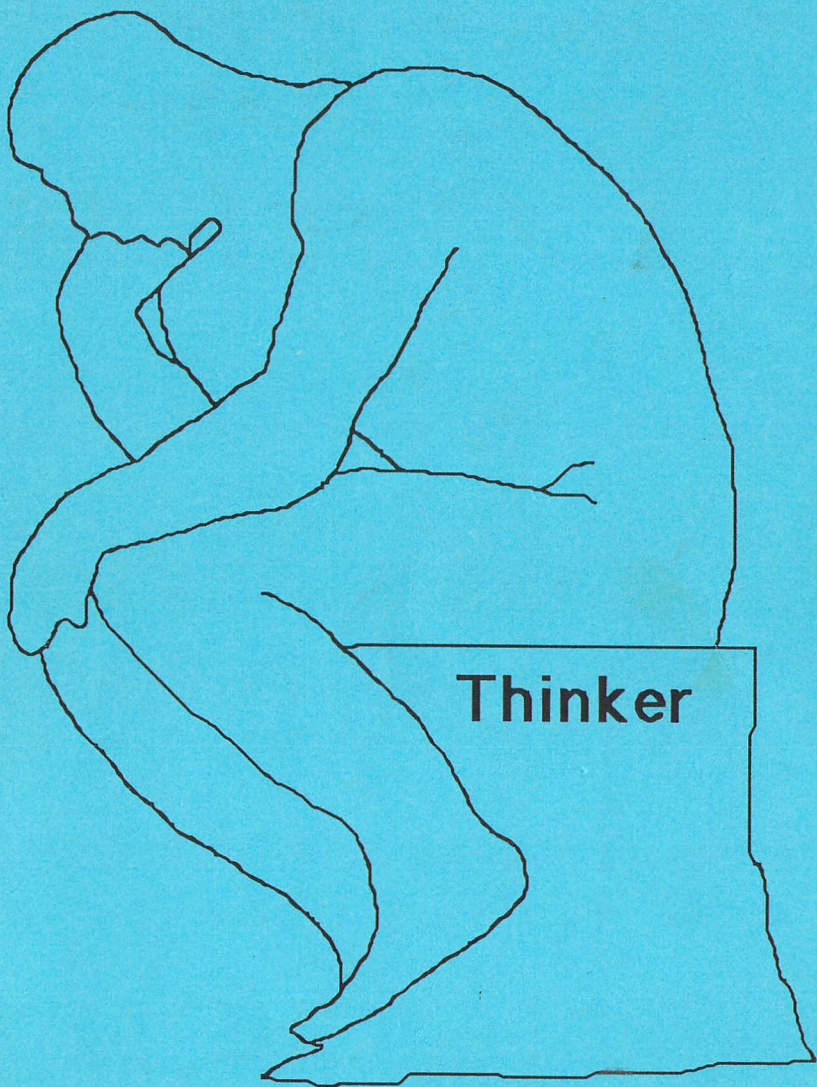




Poor Person Software

Copyright (c) 1988,1989,1990 Poor Person Software

Thinker

User's guide

Revision 2.1
November 25, 1990

Poor Person Software

**Poor Person Software
3721 Starr King Circle
Palo Alto, CA 94306
(415) 493-7234**

Copyright (c) 1988,1989,1990, Poor Person Software

Table of Contents

Introduction to Hypertext	1
Definitions	5
Thinker Documents	10
Viewing a Document	15
Editing	17
Manipulating the Structure	19
Installation	20
Getting Started	21
A Tutorial Example	23
Not So Obvious Features	27
Window Memory Buffer	27
Read Cache	28
Undo	28
Remembering Local Clip Levels	29
Power Tools	29
Number of Open Documents	29
Fast Jump Link	30
Keyboard Confirmation of Jumps	30
Keyboard Scrolling Functions	30
File Name Prefixes	30
Function Keys	31
CLI Interface	31
Option settings in the Thinker Icon	32
Picture Display Options	32
Menus Explained	33
Project	33
New	33
Open	33
Save	33
Revert	34
Save As	34
Close	34
Close Unneeded	34
Print	35
Export Text	35
Import Text	36
Get Options	37
Save Options	38

Build Index	38
Quit	39
Edit	39
Cut	39
Copy	39
Paste	39
Search	40
Next	40
Stop	40
Clipboard	40
Edit FKeys	41
Commands	42
Jump	42
Delete	45
Insert	46
Copy	48
Move	49
Sort	50
Options	50
Clip	50
Spacing	51
No Jump Confirm	51
Show Date	51
Statement Lines	51
Black on White	52
Auto Jump on INS	52
Read Cache On	52
Flag Forced	52
Hide Labels	53
Error Message Box	53
Enable See-Thru	53
Make Local Copy	54
Allow Dup Labels	55
Spell	55
Initialize	55
Spell	56
Save	56
Merge	57
Free	57
Behaviors	57
Single Click Jump	58

Allow _[;#{}) ^	58
Typeover Disable	58
Show Anchor Level	58
Reverse Video Cursor	58
Select Display Font	59
Select Jump Target	59
Default Link File	60
Replace as Typed	60
Close Unneeded Files	61
Enable Dot Commands	61
Style Delimit Links	62
Style	63
Plain	63
Bold	63
Italic	63
Underline	63
FG Pen 1	63
FG Pen 2	63
FG Pen 3	63
BG Pen 1	63
BG Pen 2	63
BG Pen 3	63
Make UPPER	63
Make lower	64
Memory	64
Gadgets Explained	65
Cancel	65
Insert	65
Delete	65
Jump	65
Clip +	66
Clip -	66
Copy	66
Move	66
Qjump	66
Zoom	66
Scroll backward	67
Scroll forward	67
Statement +	67
Statement -	67
Jump Forward	67

Jump Return	68
Warnings Explained	69
File Requester Explained	71
Sort Requester Explained	73
ARexx Interface	76
Command Port	76
ADD	77
CLOSE	77
CLOSEFILE	77
CREATE	77
CURSOR	77
CUT	78
DELETE	78
GET	78
INPUT	79
JUMP	80
PASTE	80
POSITION	80
REVERT	80
SAVE	80
SEARCH	80
SORT	80
TYPE	82
UPDATE	82
Error Returns	82
Linking to Ports	84
Macros	85
What to do If	87
Keyboard Menu Command Character Table	89
Tool Types Summary	90
License Agreement and Guarantee	92

Introduction to Hypertext

Thinker can trace its heritage all the way back to Doug Englebart's work at SRI's Augmentation Research Laboratory. By combining elements of text processing, outline processing, and hypertext, **Thinker** becomes an Idea Processor that embodies the basics of Doug's ideas for augmenting the human intellect.

What is HyperText?

Footnotes are a form of HyperText. Within the body of the text is a directive to look somewhere else in the text for more information. With footnotes the directive is a footnote number while in HyperText the directive is in the form of a character string that "names" another section of the text. In order for HyperText to work the text is divided into sections and each section has a label.

One often encounters this form of textual link when a phrase like "See Chapter 2, section 1" appears in the text. This type of link is good but can be carried further. Imagine that each paragraph in a document had a label that described the topic discussed in the paragraph. Now one could encounter the phrase "See the section <correct use of the passive voice>" which would take you directly to the paragraph on passive voice. However, where is that paragraph? What Chapter? What Page?

When the document is online and being accessed via a HyperText processor, one simply points the mouse at the phrase <correct use of the passive voice> and asks to see that paragraph. Whether it is in chapter 10 or 2 is of no concern to the reader, the HyperText processor will find it and present it to the reader.

HyperText documents do not have to be read linearly like a book. One can begin in the section with the most interest and follow the links around the document to explore the subject. The creator of the HyperText document had only to attach descriptive labels to the various sections of the document.

What is Hierarchical Text?

Hierarchical Text is an extension of outlines. Outlines are the arrangement of topics in subordinate relationships. Rain and snow are two examples of precipitation and they are properly subtopics of precipitation in an outline of a paper on weather.

If the idea of an outline is extended to an arrangement of ideas in subordinate relationships it becomes Hierarchical Text. Ideas are paragraphs or groups of paragraphs and a paragraph on precipitation might have subordinate paragraphs on the different kinds of precipitation that one might expect to find.

How Thinker combines HyperText and Hierarchical Text

Thinker is a Hierarchical Text processor with HyperText. A **Thinker** document is made up of a collection of paragraphs and images (referred to as "statements" or "branches" in this manual) arranged hierarchically and containing textual links to each other. All textual statements in a **Thinker** document can have a number of labels and can be referenced by a textual string (a link) that "names" one of the labels.

A textual statement that begins with "(" is labeled, and the labels are strings of non-punctuation characters separated by commas (",") between the first "(" and the closing ")" at the beginning of the statement. A link in the text consists of the non-punctuation characters between "<" and ">" characters anywhere in the text. When a label is a single word (or phrase with no embedded blanks), the link to it can be expressed without the "<" and ">" characters. Thus, each word in a **Thinker** document is a potential HyperText link to a statement somewhere in the hierarchy of statements that make up the document. Image statements may not be labeled.

To expand the flexibility of **Thinker**, a link may contain the name of a **Thinker** document other than the one in which the link is found. The document name is separated from the label by a comma. Thus, <Thinker:example,link> is a link to the section labeled "link" in the **Thinker** document called "example" on the disk named "Thinker".

When a statement has subordinate statements it is called a branch.

Thinker allows the manipulation of a document by moving, copying, and deleting statements and branches. When a branch is moved, all the subordinate statements are moved as well.

Display parameters control the format of the document on the screen. **Thinker** can be instructed to display only the first line of each statement and give the appearance of a simple outline. By controlling the depth of the outline visible, the user can actually "see" very large portions of the document at one time. When the document is manipulated while viewing it in outline form, all of the text and images that is part of the document are moved when the visible portions are moved.

What can Thinker "link" to?

A HyperText link can link to one of five things: 1) A statement in a **Thinker** document, 2) An IFF picture file which can be displayed in a new window (the picture is thus compressed in size and colors) or on a separate screen, 3) any Workbench application, in which case the application is launched in its own window and runs as a separate task (If a project is to be launched, there **MUST** be a project icon. If an application is to be launched, it **MUST** have a Tool icon), 4) an ARexx port, in which case a message is sent to the port, or 5) an instance of the CLI where any CLI command sequence can be issued.

The ability to link to other Workbench and CLI applications makes **Thinker** into a HyperMedia product. Drawings, sound applications, database applications, word processing applications are all possible links in a single **Thinker** document.

How does one interact with Thinker?

All interaction with **Thinker** is via the Mouse and Menu system of Intuition. There are no keyboard commands other than keyboard activation of menus. Gadgets are available for common commands and menus are available for the complete command set. Requesters are used extensively to provide a highly responsive user interface. These requesters appear directly under the mouse cursor with the cursor positioned over the most common choice.

Text on the screen is modified as in many word processors. The

cursor is positioned in front of text that is to be modified and typing is always in "insert" mode. Text is "selected" by sweeping the cursor over a block of text while holding the left mouse button down. Selected text can be "cut" from the statement and "pasted" elsewhere or simply replaced by typing. **Thinker** has "search" and "replace" functions to complete the editing capabilities.

Thinker will print a document on the preferences printer (PRT:) or prepare a text file for manipulation by a word processor or typesetting program. This manual was produced using **Thinker** and HPPrint (a Poor Person Software typesetting program for the HP LaserJet Printer). **Thinker** can also import text prepared by a word processor into a **Thinker** document.

Thinker appears deceptively simple by design. The fact that labels and links are merely parts of the text eliminates much of the complexity associated with large numbers of obscure commands. However, one should not be fooled by the appearance of simplicity. The combination of Hierarchical text, Hypertext, and Word Processing functions makes **Thinker** a powerful Idea processor. Take the time to learn **Thinker** by working with the tutorial project (EXAMPLE) found on the **Thinker** distribution disk.

NOTE: **Thinker** uses a lot of memory. One of the uses of memory is to cache blocks of documents in memory to avoid disk access. Poor Person Software recommends the use of a Floppy Accelerator such as Facc II by ASDG. The cache available in **Thinker** requires a good deal more memory than Facc II to be as effective in reducing disk access. Do not attempt to use **Thinker** on a Floppy-disk-only system without either enabling **Thinker's** read cache (the default, see "Menus Explained" under "Options") or using a program like Facc II.

Definitions

The terms used in describing a **Thinker** document have other meanings in other environments. In order to clarify the following discussion of **Thinker**, the most common **Thinker** terms are defined here.

Anchor

The statement that appears at the top of the window is sometimes referred to as the anchor statement of the window. The user navigates around in a document by changing the anchor statement. The anchor statement is changed with Jump commands.

Branch

A Branch is a statement with subordinate statements. When a statement is inserted into a document, it may be inserted at any level relative to the other statements in the document. A branch is a statement at any level that has statements below it. A branch may have a branch subordinate to it. A Statement with no subordinate statements is sometimes referred to as a Branch.

Clipping Level

Clipping level refers to the number of document levels that are displayed in a window. A window with a clipping level of 2 will show only the two highest level statements. Clipping can be used to view a hierarchical document at different levels of detail. The document can be manipulated with clipping levels that exclude much of the document from the screen. All of the invisible text is moved with the visible text.

The clipping level can be chosen for the entire document or for single branches. The Clip gadgets ("+" and "-") at the top of the screen or the Clip menu sets the clipping level that applies to the document as a whole. "+" or "-" gadgets in front of statements manipulate the clipping level applied to the single branch.

Group

A group is a selected contiguous sequence of branches at the same level.

Jump

When the anchor statement of a window is changed, a different portion of the document becomes visible in the window. This is referred to as a jump. Jumps can be made relative to a statement visible on the screen or relative to a statement named in a link. It is also possible to jump to picture files, Workbench applications, ARexx ports, and CLI instances.

Label

Any textual statement in a **Thinker** document may have labels (Image statements do not have labels.) A label is a name by which the statement is known and is like a key in an indexed file. **Thinker** recognizes a statement that begins with "(" as its first character as being a labeled statement. The labels are substrings of all of the non-punctuation characters between the initial "(" and a trailing ")" separated from each other by commas. "(These,are,labels)" is a label string with 3 labels. The case of letters in a label is not significant and only the first 19 non-punctuation characters of each label in the string are significant. There is an option to make some of the punctuation characters significant in labels.

Labels in a **Thinker** document need not be unique. Any request to jump to a label that is not unique will cause a requester to appear which displays the first characters of all statements with the non-unique label. The user points at and clicks on the desired statement to select the target of the jump.

Level

Statements are arranged in a hierarchy. The most important statements are at level 1. Level 2 statements are subordinate to level 1 statements as in an outline. **Thinker** does not limit the number of levels in a document but the display on the screen will be less than satisfactory for more than about 30 levels.

Link

A "link" is the appearance of a label in the text. A link is either a single word or a string of characters delimited by "<" and ">" (An option exists to delimit links with changes in text style.) A link consists of three parts.

Part one is optional and names a project, **Thinker** document or file.

Part two is separated from part one with a "," and names a branch. Picture file links consist of only the file name and the comma as in <Thinker:art/mirage,>.

Part three is optional and is separated from part two with a ".". The string that follows the "." is a sequence of single character commands that continue the Jump relative to the target label (Up Down, Successor, Predecessor, Tail of a Plex, or highlight a string within the target statement.) The link <example,insert.dtfdown> is an example of a link to a statement with the label "insert" in the file "example", then moves "Down" to the first subordinate statement, then moves to the "Tail" of the plex and then "Finds" the string "down" and highlights it.

Links may be typed into requester strings as the address of statements when certain operations invite the user to designate a statement (only the Jump Link Address requester allows the use of "." command extensions). Links typed into requesters do not have the surrounding "<" and ">" characters. Links can be activated by putting the cursor over the link in the text and double clicking the left mouse button or by using the Jump Link command.

Links may designate labeled statements in the same or different **Thinker** document, IFF picture files, any Workbench application such as a music player, paint package, or CAD application, an ARexx port, or an instance of the CLI so that any CLI application can be invoked. Use the file name or program name in the file portion of a link followed by a comma to specify a link to an application.

Mark

Thinker sometimes invites the user to designate a statement as the target of an operation such as "delete" or "move". The user may elect to mark the statement on the screen or type in a link that designates the statement. A statement is marked by moving the Pointing Cursor (a hand) such that the hand points to the statement and pressing the left mouse button.

Origin

The first statement in a document is referred to as the origin statement. The origin statement may be moved or deleted in which case some other statement becomes the origin statement.

Plex

A complete group of statements. All of the subordinate statements in a branch constitute a plex. All the first level statements in a document constitute a plex. A plex is designated by pointing to any member of the plex.

See Through Link

Links can be used to extract statements from one document into another. The statement named in a "See Through Link" can be displayed on the screen in place of the link. This allows a **Thinker** document to be a composite of other **Thinker** documents. Details about the nature of "See Through Links" are found in the section of the manual describing the "Enable See-Thru" menu selection (under "Options").

Statement

A statement is a block of text or an image. Textual statements are like paragraphs in many word processing programs. **Thinker** does "word-wrap" within a single statement but not across statements. Text selection (by sweeping the cursor over text with the left mouse button held down) is limited to text within a single statement. Statements may contain up to 2000 characters of text including standard and extended characters from the Amiga keyboard. The <ESC> and

carriage return characters are also allowed.

View Specifications

The appearance of the text in the window is controlled by three parameters that together are called the view specifications. These parameters are "Clipping level", "Spacing", "Number of Lines" and are each explained in the Options Menu section.

Window

Only a small portion of a **Thinker** document can be displayed on the screen at any time. **Thinker** uses Intuition windows to view portions of a document. Up to 8 windows may be open at any time. Each window may display a portion of a different document or different portions of the same document.

Thinker Documents

Thinker documents are not always meant to be read linearly. While this manual was created using **Thinker**, many uses of **Thinker** take more advantage of the Hypertext and Hierarchical text features. A skilled **Thinker** user creates documents with a definite flavor. These documents do not always read well when flattened and printed on paper and are meant to be read online.

Outline nature of **Thinker** documents

If a **Thinker** document is viewed with a clipping level of 1, only the first level statements (often these statements are topic names or section headers) will be shown. It is rare but not impossible to have blocks of text at this level. An executive summary is presented with the clipping level set at 2. As the clipping level is increased, more and more details are revealed. If **Thinker** is used to design a program one would expect to find code fragments or pseudo code at the deepest levels of the document.

Hypertext features

Information is only presented once. A reader should be able to start reading a **Thinker** document at the topic of most interest. All references to terms that require a definition in the context of the document should always have a link to the section of the document where the term is defined. The fact the **Thinker** will treat any word as a link means that a Glossary section could define all terms and contain links to the sections that provide further clarification.

When a new term is encountered, the reader double clicks on the word to open a window over the Glossary section defining that term. If there is more information available then the Glossary entry will contain a link to the section of the document where the term is introduced and described in detail. Double clicking on the link can move the Glossary window to the detailed information.

These links make **Thinker** ideal for online help where the reader can go directly to the section of interest and still be able to locate quickly all the pertinent information.

See Through Links make it possible to automatically extract information from one document into another without double clicking on the link. A new Thinker document might summarize the information in many other documents by using See Through Links. When the information in the original documents is updated, the new summary document will display the changes.

Planning a book

One possible use of Thinker involves coordinating all the details of a novel. One technique that might be of value is to have several major sections of a Thinker document for various aspects of the planning process. One section would have a labeled branch for each character in the book. Other sections would describe historical events of interests and descriptions of each location. Each of these sections would have labeled statements for each detail.

The section of the document that describes the story line (plot) of the book would contain links to the sections containing detailed information. The plot can be reviewed without having to wade through details that might confuse the issues. As needed, each reference in the plot to details about characters, scenes, events, etc. could be checked by jumping to the section of the document where this information is found. Multiple windows help organize the thought process.

Small Example:

(Characters)

(Anastasia) - The heroine

Lots of stuff about Anastasia

(Scrully) - The hero

Lots of stuff about Scrully

(Places)

(San Francisco) - Home of Anastasia

Lots of stuff about SF

(San Jose) - Home of Scrully

Lots of stuff about SJ

(New York) - Place of story

Lots of stuff about New York

(Plot)

Anastasia meets Scully in <New York>
They marry and raise a family
The End

Organizing picture files

Since links may be to IFF picture files, one use of **Thinker** might be to build a catalog of picture files. Each section of the document would have a first level statement with a label indicating the type of pictures indexed. The subordinate statements would contain descriptions of the picture and a link to the picture file. Pictures are located by jumping to the section of the document with the descriptions for the correct type of picture and then double clicking on the link to the correct picture.

Small Example:

(People)
 (Lauren) <Thinker1:art/lauren,>
 (Bob) <Thinker1:art/bob,>
(Birds)
 (Bluebird) <Thinker1:art/bluebird,>
.....

Thinker documents as databases

Labeled statements in **Thinker** documents are much like indexed records in a database. There is a great deal of flexibility above the typical database, however. Records are not a fixed format. Records can contain a variable number of pointers to other records. Records can be part of a document meant to be read.

Using **Thinker** to write is like writing inside a database as references can be checked with the click of a mouse button. Using **Thinker** is like having a completely free form database with references to applications, pictures, and documents.

A Small Example:

(Distributors)
 (American Software) - Address
 (Dealers)
 (Winner's Circle) - Address
 (HT Electronics) - Address
 (Unregistered buyers)
 (Nymous-A) - Address
 Orderdate:
 Shipdate:
 (Registered buyers)
 (Huxtable-P) - Address
 Orderdate:
 Shipdate:
 Amount:
 Registration:

Thinker as a desktop organizer

A desktop document can be used as an alternative to the Workbench. Within the document are labeled statements covering various working task areas such as Letter Writing, Programming, Finance, etc. Subordinate to these statements are labeled statements for each task in that area. These statements describe the task and may include sentences describing the progress of the task. Finally, there are links within the statements that name the Project or Tool used to work on the task. A description of a program might contain the name of the source for the program so that you can link directly to the source and work on it within your favorite editor. Other statements can link to other Projects and Tools, any of which can be launched in multitasking mode.

A Small Example:

(new projects)
 (Thinker) - Learn thinker <Thinker:example,>
 (old projects)
 (whoopydo) - Work on favorite program <Source:whoopy.c,>
 (appointments)
 (Monday) - Meet Alice at Restaurant

Screen real estate

Thinker is designed to view and manipulate large documents on a small screen. Careful use of clipping levels and the statement lines makes it possible to "see" large portions of a document on a 24x80 screen. Selective expansion of individual branches and the use of multiple windows maximizes the use of the limited size screen. **Thinker** senses the screen size during initialization and adjusts to any screen size supported by the Amiga including extra large screens available with additional hardware.

Viewing a Document

A **Thinker** document is a hierarchy of statements. There are three view specifications that control what portions of this hierarchy is visible on the screen.

Clipping level

The clipping level controls the depth of the hierarchy that is displayed. A clipping level of 1 displays only the first level of the hierarchy (the major topics) and as the clipping level is increased more details are displayed.

There is a global clipping level controlled by menu options and the Clip gadgets at the top of the screen. Each branch has a local clipping level controlled by gadgets in front of the statements. Local clipping levels are always higher than the global levels.

A document can be searched rather quickly by starting with a global clipping level of 1 and using the local clipping gadgets to expand the areas of interest.

Number of lines shown in each statement

The document can be viewed as an outline by setting the view specifications to display only the first one or two lines of each statement or the full document can be viewed by setting the view specifications to display all lines in each statement.

Spacing between statements

The spacing can be adjusted to maximize the amount of material on the screen or to maximize the readability of the material.

One views a linear document by scrolling over the document. One views a non-linear **Thinker** document by moving the window to particular places in the document with the Jump command. Jumps can either be relative to the document structure or can use links.

Structure relative jumps include moving the window up and down in

the hierarchy or forward and back at the same level.

Links provide the most flexible way to navigate around a set of **Thinker** documents. A link in the text ties the displayed text directly to text elsewhere in the document or in another document. Links may be typed into a string gadget or found displayed on the screen. If a link is displayed on the screen merely double clicking the cursor over the link executes the jump. Any single word in a **Thinker** document can be a link! Any phrase that is displayed in a text style other than plain (**bold**, *italic*, underlined) or with colors other than the default can be a link!

For those times when it is desired to scroll linearly through a document several scrolling gadgets are provided. Scroll forward can be by single, single statement or full screen. Backward scrolling is by single statement or to the previous position. There are Jump commands that jump forward by a full screen or back by a statement. It is worth remembering that **Thinker** documents can be much larger than the Amiga Memory and only the active portions of the document are in memory. This and the fact that **Thinker** documents are randomly accessed files makes it impossible to provide scroll bars.

Scrolling is also possible using the cursor keys in combination with Shift and Alt. Alt-Up and Alt-Down scroll the document up and down by single statements. Shift-Down scrolls forward by single lines.

Editing

Text within a statement is edited much like text in a standard word processor.

The cursor is positioned by using the mouse or cursor keys. The cursor keys move the cursor within a statement, between statements, and will cause line by line scrolling forward or statement by statement scrolling backward. Shift-LeftCursor puts the cursor at the beginning of the line, Alt-LeftCursor puts the cursor at the beginning of the statement. Shift-RightCursor puts the cursor at the end of the line (this is the same as the beginning of the next line), and Alt-RightCursor puts the cursor at the end of the statement.

Alt-UpCursor and Alt-DownCursor scroll the window backward and forward by one statement. Shift-DownCursor scrolls the window forward by one line. There is no scroll backward by one line.

When the mouse is used to position the cursor, the left mouse button is used to mark the place in the text. Subsequent typing inserts characters at the cursor location. "Backspace" deletes a character to the left of the cursor and moves the cursor to the left. "Delete" deletes a character to the right of the cursor and does not move the cursor.

All printable keyboard characters are allowed in text (including alternate characters and deadkeys). In addition the <ESC> character and <CR> character are allowed. The <ESC> character is useful for imbedding printer commands in text. The <CR> (carriage return) character is used in formatting tables as a single statement rather than as a group of statements. The <CR> character is visible as "~" and causes the cursor to move to the next line of the statement in the first column of the statement. A <New Line> character is put into the text.

A block of text may be selected by holding down the left mouse button while moving the cursor over the text on the screen. Only text that is totally within a single statement can be selected in this fashion. Once selected, the text can be "Cut" from the statement and "Pasted" at some other location. Selected text is replaced by any typed characters and the selected text is forever lost.

Undo is supported with the "Revert" menu selection. All modifications including structural changes are deleted. The user may select "undo points" by using the "Save" menu selection. In fact no changes are recorded on the disk until Save is selected. There is a reminder warning presented if the user attempts to leave Thinker without a Save. The requester gives the opportunity to save or discard the changes.

Search and Replace operations are selected from the Edit menu.

Manipulating the Structure

Statements in a **Thinker** document may be moved within the hierarchy, moved to another document, copied to other places in the same document, or copied to other documents or deleted. The movement, copying, or deletion of statements does not require that the entire structure to be moved, copied or deleted be visible on the screen as parts of the structure may be obscured by the selection of viewing specifications.

It is the ability to manipulate whole structures of the document that make the hierarchical text such a powerful tool. The viewing specifications can be set to view the document in its outline form (set the number of lines for each displayed statement to 1) and the entire document can be restructured.

Move and Copy and Delete options are available for individual statements, branches, and groups of statements or branches. In all cases a sequence of requesters walks the user through the operation. Some of these requesters have a "Previous" gadget which will recall the previously typed string gadget. Labels may be typed into many of these requesters to designate a statement or the statement may be selected with the mouse.

The structure manipulation commands can be initiated from the menus, from the gadgets at the top of the window, and from the "power tools" requester. The "power tools" requester is initiated by double clicking on the right mouse button. From this requester it is possible to directly insert a statement or initiate a Move, Copy, Delete, or Jump command.

Installation

Make a Copy of the Master diskette.

Decide where you want to put the **Thinker** program. In a two drive system it might be best to leave **Thinker** on its own disk. You can then simply insert the **Thinker** disk any time you want to use **Thinker**. You can create new Drawers on this disk to organize your documents. On a single drive system you will want to make a new Bootable disk and put **Thinker** on that disk and name the disk "THINKER2". After creating the new Bootable disk (using the **INSTALL** program or duplicating an existing Bootable disk) drag the **THINKER** Icon, the **THINKER.LEX** icon, the **ART** Drawer icon, and the **EXAMPLE** Icon from the **THINKER2** disk to the new disk. Do not put these Icons in a Drawer or you will have to use **INFO** to modify the **EXAMPLE** Icon and the **MIRAGE** icon in the **ART** Drawer. Finally, rename the new disk to **THINKER2** and boot from it.

The **Thinker** document **EXAMPLE** has an Icon with a Default Tool Name of "THINKER2:THINKER". This document is used in the Tutorial section of this manual.

Thinker makes use of "double clicking" to provide short cuts to some operations. You will probably want to use **Preferences** to set the mouse button double click time to a very short value.

A suggested hard drive installation is to create a directory called **THINKER2** and copy the entire contents of the installation disk into this directory. Modify your startup sequence to include the following assign command.

ASSIGN THINKER2: HDn:THINKER2

Where "n" is the drive number (0, 1, 2, etc.)

Getting Started

Running from the CLI

The command **THINKER [PROJECT-NAME]** is used to start **Thinker** from the CLI. If a project-name is specified, the **Thinker** window is automatically positioned at the origin of the document. If the project name specified does not exist or is not a **Thinker** document or IFF picture file or if no project-name is specified then the **Thinker** window is blank. Use a Jump Link command or select the New or OPEN menu items to open a **Thinker** document.

The **-NOW** or **-NOWINDOW** option on the **THINKER** command is used when **Thinker** is to be used as a database in an ARExx program. With the **-NOWINDOW** option specified, **Thinker** will not open a window and can only be communicated with by its ARExx port (named "Thinker", note the mixed case.)

Running from the Workbench

Double clicking the **Thinker** Icon will initiate **Thinker** with no active **Thinker** document (a blank window). The Open menu, Jump Link command or New menu must be used to open a **Thinker** document. If a document Icon is double clicked then **Thinker** is started with the window positioned at the origin of the document.

Thinker terminates when the last window is closed, when the "Quit" menu option is selected, or when it receives a CLOSE or QUIT command on its ARExx port.

Document names and AmigaDOS files

There is always the question about where **Thinker** will put a newly created document and what the Default Tool Name of the new Icon will be. The Drawer name for a new document can be specified as part of the name of a new document by including a ":" in the name. If a Drawer is not named then the document is put into the Drawer from which **Thinker** was launched.

If the **Thinker** Icon was double clicked, the new document will go into

the Drawer that contained the **Thinker** Icon. If a **Thinker** document Icon was double clicked, the new document will go into the Drawer that contained the document Icon.

The Default Tool Name of the new Icon will designate the exact place that the Workbench found the **Thinker** program. If you move the **Thinker** program, you will have to use INFO to modify the Icons for all **Thinker** documents.

Backup

Thinker documents are easily backed up from the Workbench by dragging their icons into the window of a backup disk. Since a **Thinker** document is a single file, backup from the CLI is accomplished by copying the document to a backup disk.

COPY Volume:drawer/file to Backup:drawer/file

Backup from within **Thinker** is accomplished using the "Save As" menu item in the "Project" menu.

A Tutorial Example

The first part of the tutorial will walk the user through the process of building a **Thinker** document. The second part of the tutorial makes use of an existing **Thinker** document called "example". In the following examples do not type the "" characters.

The **Thinker** document "Example" is a tutorial that leads the user through a short example of using **Thinker** to view and manipulate a document. Duplicate the EXAMPLE document using the "Duplicate" option of the "Workbench" menu before trying the tutorial. If you want to recover the original EXAMPLE document you can discard the EXAMPLE icon and "Rename" the "copy of example" icon.

Start the first part of the tutorial by double clicking on the **Thinker** icon. This will start the **Thinker** program and bring up a window with the basic gadgets and menus but will have no document displayed. At this point we might jump into some existing document or create a new document. One would jump into an existing document with the Jump Link command and create a new document with the Create New menu selection (in the Project menu).

As an example of jumping into an existing document, there is a section in the "example" project that explains the format of a link. The label on this section is "link". To jump to that section, begin by selecting the Jump Link command (either from the menu bar or by selecting the "Jump" gadget at the top of the window and then selecting the "Link" gadget in the Jump Requester.) Next, type in the string "example,link" (without the " marks) followed by typing the "RETURN" key. Finally, select the "This Window" gadget in the Jump Confirmation Requester.

Typing the string "example," (note the comma but no label typed after the comma) would jump to the beginning (origin) of the document "example".

Whether or not you take the diversion suggested by the previous paragraphs, you create a new **Thinker** document by selecting the "New" menu option from the "Project" menu. Do so now. Note that a requester pops up inviting you to enter a project name. The first part of the requester is filled in with the name of the current Drawer. Since you

double clicked the **Thinker** icon directly, this Drawer name will be the Drawer in which the **Thinker** program was found. The Requester is already activated so simply type in the name of your new project. In this example we use the name "tutorial". If you entered the name "example" you will be greeted by a requester asking if you want to replace the "example" project that already exists. You should respond by selecting the CANCEL gadget and begin again typing the name "tutorial".

After some disk activity the window will display the new document (note the change in the window title) and the document will contain a single statement (a Copyright notice). The first thing to do is to replace the Copyright notice with the first statement of our new document. You might try using the Delete (Branch) command but you will find that you cannot delete the only branch in a document.

Move the pointer over the Copyright notice and observe that while the pointer is within the statement it is shaped like a cursor. Move the cursor shape to the very beginning of the statement and press the left mouse button. While holding down the left mouse button, move the cursor to the end of the statement. Notice that the selected text becomes highlighted as you move the cursor. When you have the entire statement selected, release the left mouse button. Type in the string "What I did last Summer" (do not type the RETURN key). Notice that the selected text is removed and replaced with the typed string.

Now we are going to add some statements. This initial document will look a great deal like an outline. Remember that any statement you type could be as long as 2000 characters and fill the entire window. In order to keep this first example short, we will only type a few words in each statement.

Select the "Insert" gadget at the top of the window. Note that the pointer changes into a hand with a pointing finger. Move the hand so that the finger points at the first statement. Press the left mouse button and notice that an Insert Confirmation Requester pops up with the pointer pointing to the word "After". If you wanted this new statement to be directly after the first statement, you would simply press the left mouse button again. Instead, move the pointer over the word "Down" in the requester and press the left mouse button. Notice that the text insert cursor appears on the screen just below the first statement and indented by two characters. Type the string "(school)Went to Summer touch typing

classes."

Select the Insert gadget again and move the hand to point to the statement you just typed. Press the left mouse button twice (the first time selects the statement and the second time selects the word "After" in the confirmation requester.) Type the string "(beach)Went to the beach with Bob and met his little girl Lauren." (Who is Bob?) Bob is someone that attended the same typing class so you will want to put any information about Bob in statements that are subordinate to the statement labeled "school". Position the pointer over the first statement (the one labeled "school") and double click the right mouse button. The Insert Confirmation Requester should pop up and you should select the "Down" gadget. Type the string "(bob)Met Bob, a computer science student who wanted to learn typing skills."

Move the pointer back to the last statement "(beach)Went...." and position the cursor over the word "Bob" and double click the left mouse button. Notice that the Jump Confirmation Requester pops up with the pointer positioned over the words "This Window". Move the pointer over the word "New Window" and press the left mouse button. Notice that the new window opens with the statement "(bob)Met Bob,..." as the anchor of the window. This is an example of a hypertext link. "Bob" is the label on the statement "(bob)Met..." and the double click of the left mouse button with the cursor over the word "Bob" was a Jump Link command. The same effect could be caused by selecting the "Jump Link" command from the "Commands" menu, or by selecting the Jump (Link) gadget at the top of the window and typing the characters "bob" followed by RETURN.

Who is Lauren? Position the cursor right after the word "Lauren" and press the left mouse button once. The text input cursor should appear between the "n" and the ".". Type the string " (See <Thinker2:art/lauren,>)". This is a link to a picture file. Note that the comma must be present to indicate that the link is to some other file and not the document shown in the window ("tutorial").

You have just added a hypertext link to an IFF picture file. This link had to be surrounded by "<" and ">" because it contained punctuation characters (":" and ","). Move the cursor between the "<" and the ">" and double click the left mouse button. When the Picture Confirmation Requester pops up select the "Full Screen" gadget. After a pause, a picture of Lauren appears (From the NewTek "Digi Paint" package.)

Typing any character or pressing the left mouse button will remove the picture and return the **Thinker** screen. Repeat the selection of the picture link and select the "New Window" gadget. This time an approximation of the same picture will be drawn in a new workbench window that you may move around on the screen. This approximate picture is drawn with the 4 colors available on your workbench. The window has an invisible close gadget in the upper left corner that will close the window when selected.

If you want to save the changes you have made to this new project, select the "Save" menu selection from the "Project" menu. "Revert" will restore the project to its initial state (with the Copyright notice).

This completes the first part of the tutorial. The second part of the tutorial uses an existing document called "example". You can begin by closing the **Thinker** window (select the close gadget in the upper left corner) and starting all over again by double clicking on the **EXAMPLE** icon, or by doing a Jump Link to "example," (don't forget the comma). In either case you should read the next paragraph before going on with the second part of the tutorial.

Note: When a file name is selected in the file requester, the comma is optional unless you wish to add a label.

Start the second part of the tutorial by double clicking on the **EXAMPLE** icon or by selecting the icon and use the "Open" option of the "Workbench" menu. Read carefully and follow the instructions. It is helpful if you read ahead a little before executing the instructions. The tutorial is not very long or complicated but is intended solely to give the reader a little familiarity with the behavior of **Thinker** and build a little confidence for the first time user of **Thinker**.

Not So Obvious Features

Most of the operation of **Thinker** can be discovered by exploring the menus and selecting the gadgets. There are, unfortunately, some behaviors that are not manifest by this exploration. There are also some curious restrictions. Poor Person Software tries hard to produce software that is useful and complete but without all the frills that would make it expensive.

Restrictions

The maximum number of open files is determined by the **MAXFILES** **ToolType** in the **Thinker** icon.

When the **Mark** option is used for both the beginning and end of a group, both ends of the group must be in the same window, and must be in the window used to start the operation.

Only 19 characters of a label are significant. Label strings and link strings (the string between **< >**) are limited to 256 characters. Each label in a label string is truncated to 19 characters.

Only the first 32 statements with duplicate labels can be reached via a **Jump Link**.

Features

Window Memory Buffer

When statements are displayed in a window, the text information is in a memory buffer where it is easy to update the data based on the keyboard input. Any time that a **Jump** command is executed, "**Save**" is requested, or another project is opened, the changes in the memory buffer are moved into the **Update Pool**. When there are changes in the **Update Pool** a ******* character appears in the title line in front of the filename string. Since the **Window Memory Buffer** is not moved into the **Update Pool** at each keystroke, the ******* will not appear immediately upon making a change in the window. Selecting the window sizing gadget will write the **Window Memory Buffer** into the **Update pool**.

The presence of the Window Memory Buffer is only important when there are multiple windows open over the same document. Changes made in one window are not visible in another window until the Window Memory Buffer for the changed window is written into the Update Pool. The previous paragraph contains a discussion of when the Window Memory Buffer is written into the Update pool.

The Update Pool is where modifications to a **Thinker** document are stored until the user selects either "Save" or "Revert". After one of these selections the Update Pool is emptied.

Read Cache

Thinker keeps a cache of the all modified portions of a **Thinker** document in its memory (the Update pool) but only a small portion of the document that has been read from the disk is kept in memory (a read cache). Since **Thinker** documents are randomly accessed AmigaDOS files, updating the screen could be a painfully slow process without this read cache. The read cache is small and can be turned off if some Floppy Accelerator like Facc II from ASDG is used. ASDG did such a good job that Poor Person Software did not feel that it was worth the time to include much of a read cache in **Thinker**. The read cache is large enough to insure that screen updates are reasonably fast.

As an alternative, small documents can be copied into the RAM: device and manipulated there. When you are finished and have saved all your changes, copy the document back to a disk.

Undo

Once modifications have been applied to a **Thinker** document they cannot be "undone". However, **Thinker** will accumulate all the updates to a document in its memory (the Update pool) until "Save" is selected in the Project Menu. The user should learn to schedule updates to allow for backout of recent errors.

Remembering Local Clip adjustments

When the clipping level for a branch is adjusted using the "+" gadgets in front of statements, this action is recorded in a small in-memory data base. This database has room for 32 selections of "+" gadgets in each project. If the 33rd entry is made into this database, the oldest entry is purged.

Power Tools

Double clicking the right mouse button will cause **Thinker** to enter "power tool" mode. An insert confirmation requester and a set of tool gadgets will appear and invite the user to select a relative level to insert a new statement (as in the last step of the insert command) or to select a Jump, Move, Copy or Delete tool. If the pointer is within a statement (showing the text pointer) and the user selects one of the levels, a statement will be inserted. If a tool gadget is selected **Thinker** behaves as if the tool gadget at the top of the window was selected.

The number of open documents is limited by a ToolType in the **Thinker** icon or a default of 16.

When **Thinker** jumps to a new document, it must open a new AmigaDOS file. Since there may be outstanding updates on the document that was jumped from and you may soon return to that document, the document is not closed.

The number of open files may be increased by updating the **Thinker** icon and changing the MAXFILES=nn ToolType. About 800 bytes of memory is reserved for each possible open file.

There are three ways to close files. The "Close" menu selection will close the file visible in the active window. The "Close Unneeded" menu selection will close all files that are not displayed in any window and have no outstanding updates. This is a good way to close files after a series of Jump Link commands. The third way is to select the Close Unneeded Files behavior. This selection closes files as they become inactive.

Fast Jump Link

Double clicking the left mouse button with the cursor positioned over a link (or word or style delimited phrase) is equivalent to doing a Jump Link command with the Mark option and positioning the hand over the link (or word or style delimited phrase).

Keyboard Confirmation of Jumps

In all cases when the confirmation requester is brought up to complete a jump. The default confirmation can be effected by typing the RETURN key on the keyboard. NOTE: when the cursor is close to the edge of a window when a requester pops up, the requester may not be positioned so that the cursor is over the default option. If you start a jump link sequence by selecting the Jump gadget at the top of the screen and don't move the cursor before the Jump Confirmation requester pops up, the requester will be crammed to the top edge of the window and the cursor will be positioned over the "New Window" selection. The default selection that the Return Key activates is still the "This Window" selection.

Keyboard Scrolling Functions

Besides the 5 scrolling gadgets explained in the Gadgets section combinations of the Up and Down cursor keys and the Shift/Alt keys cause scrolling. Alt-Up scrolls toward the origin by one statement. Alt-Down scrolls toward the end of the file by one statement. Shift-Down scrolls toward the end of the file by one line.

File Name Prefixes

For some applications it is awkward to have the full pathname of files as part of links. If the file is moved from one floppy disk to another or from one directory to another, then all the documents which contain the link must be changed. The use of file name prefixes alleviates this problem.

The CLI environment has a mechanism called ASSIGNS that solves this problem for CLI programs. In fact ASSIGNS can be

used for **Thinker** links. A link of the form <ABC:filename,label> would look on the floppy disk with the label ABC: or if there were an active ASSIGN of the form "ASSIGN ABC: HD0:xyz" would look on the hardisk in directory "xyz".

Thinker has a similar mechanism. You can use INFO to add file name prefixes to the **Thinker** icon (do not put file name prefixes in project icons). These prefixes act similarly to ASSIGNS. If there is a tooltype of the form "ABC:=HD0:xyz/" then links of the form <ABC:filename,label> would be resolved as "HD0:xyz/filename,label". The string on the right side of the tooltype is substituted for the string on the left side when found at the beginning of a link file name.

Function Keys

If you have ARexx installed on your system, the F1 through F10 keys can be used for macros. The Edit Fkeys menu is used to edit the command strings that these function keys represent. The command strings are sent to **Thinker's** ARexx port when the key is pressed. The **Thinker** ARexx port interprets these commands. Any command not understood by **Thinker** is sent to the ARexx master process and executed as an ARexx macro.

CLI Interface

The special link <CLI,command> uses ICONX to execute the "command" command in the CLI environment. "Command" can be any legal CLI command or set of commands separated by line feed characters. Line feed characters are entered at the keyboard while typing into a statement using the "Return" key and show up on the screen as ~. When typing a CLI command string into the Jump Link string gadget, type the "~" character to indicate Carriage Returns.

The ICONX program must be in the C: directory for this special link to work.

Option Settings in the Thinker Icon

Any option or behavior setting can be stored in the **Thinker Icon**. In this way the user may customize the default environment. These option settings are put into the icon using the **INFO** menu selection of the Workbench. Look at the format of the ToolType strings in a **Thinker** project icon after a **Save Options** selection and copy the string exactly. The format of these strings is very rigid. The **WINDOW= ToolType** is not honored in the **Thinker** icon.

Picture Display Options

There are four options for the display of IFF picture files. Besides the "Full Screen" and "Workbench" option there are two options with enhanced colors for the Workbench. Both of these new options use a color dithering technique to simulate 10 colors on the Workbench using combinations of the 4 colors available. The resulting 10 colors will probably not cover the complete spectrum of colors as few workbenches have all the primary colors in the palette. There is a correction in the dither algorithm for the usual lack of green in workbench palettes.

"Full Screen" displays the picture as the artist intended. HAM and OVERSCAN are supported.

"WB" is a 320x100 size workbench window in four colors.

"WBx" is an enhanced color display in the 320x100 size. It takes about twice as long to display this window as it does to display a "WB" window.

"Xfer" is an enhanced color display with a size of 160x50. This window size is probably the size that most people will use to insert images into documents (it takes 2000 bytes, the maximum size of a text statement). Even with the enhanced colors, these windows are displayed fairly rapidly.

Menus Explained

Note that in Release 2.0 some of the Menu names and Menu keyboard command characters have been changed to conform to Amiga Standards.

Project

These menu selections manipulate **Thinker** documents.

New

A new **Thinker** document is created. The name of the document is typed into the string gadget of the "Enter Project Name" gadget. If the name of the project is to include a Drawer name, the ":" must appear in the name. Typing the <carriage return> key activates the creation of the document.

If the project name is not unique the user will be asked (via a requester) whether or not to replace the existing project. Replacing the existing project will cause all data in the replaced project to be lost. You cannot replace a file that is being used by **Thinker**.

The new project is initialized with one statement (a copyright notice) and **Thinker** opens a new window to display the new file. If there is no file displayed in the active window then **Thinker** will Jump to the new file in the active window.

Open

Open is equivalent to the gadget sequence: Jump, Link, File Requester. A file requester is brought up and the selected project is opened and displayed as if you had Jumped to it via a link.

Save - RAMiga S

All modifications to the document that is shown in the active window are applied to the project on the disk. Once this is done the modifications may not be removed. Until this is done the

project on disk remains unchanged.

Note that only the modified portions of the document are saved. "Save" takes only a few seconds unless a large number of changes have been made.

Revert

All modifications to the document shown in the active window since the last "Save" or since the document was opened (via Open or Jump Link) are discarded. The document "reverts" to its previous state.

A confirmation requester reminds you of the potential loss of work that selecting this menu item might cause.

Save As

A new document is created much as in New. The document that is displayed in the active window is copied into this new document and all the updates are applied to this new document. An automatic jump to the new document is executed. The document in the active window at the time of the Save As is left in its original form (without the updates) as if Revert had been selected.

Close

Only 16 projects may be open at one time (or the number selected with the MAXFILES= ToolType.) It may be necessary to close a project in order to allow a new project to be opened. The Close menu item closes the project that is displayed in the active window and displays a blank window. The user must do a Jump Link command, New menu selection, or Open menu selection in order to display a document in the window.

Close Unneeded

All files that are not displayed in a window and have no outstanding updates are closed. This action frees up entries in the open file table .

Print

The document visible in the active window is sent to the PRT: device with the same view specifications that are set in the active window. The sub-menu items allow the entire document to be sent to the PRT: device (the "Project" sub-menu option) or only the branch that is the anchor statement of the active window (the "Branch" sub-menu option).

Text styles are translated into Printer Preference sequences but colors are not represented. The line width of the printer is selectable using the EXPORT requester.

Export Text

The document visible in the active window is written into an AmigaDOS file as a flat ASCII text file. The name of the destination file is typed into the first string gadget in the "Enter File Name" requester. The formatting performed on the text file is determined by the prefix format typed into the Indent Prefix string gadget in the same requester.

The Indent Prefix is actually a format string for a PRINTF call in the C language. The number of indent positions for the statement is the variable that is to be printed according to the format string (usually using the d format conversion for an integer). The statement below illustrates how the indent prefix is used.

```
printf("indent prefix",indent#);
```

The default format string for the Indent Prefix is \in d\ which will prefix each statement with a string that is interpreted by HPPrint as an indentation setting. The Indent Prefix string should be modified to produce the command required by your favorite print formatter or word processor.

Two special strings are possible in the Indent Prefix. First, the NULL or empty string (delete all the characters in the prefix string gadget). If the Indent Prefix string is NULL then the ASCII file is formatted without any indentation at all. Each

statement is put into the ASCII file as a continuous string of ASCII characters.

Second, the single character "!" causes the ASCII file to be formatted as in the Print option. Each line is indented the amount shown on the screen and ends with a new line character. The appearance of the text in the file is the same as in the window. The line width of the output is selectable using the "Line Width" gadget. This line width also controls output from the "Print" menu.

The Line Width gadget in the Export Requester selects the length of the text line for word wrap in the ASCII file for both the Export and Print functions.

The Bold, Italic, Underline gadgets select style characters that will surround text of the corresponding style. If you wish style information to be discarded, put null strings in these gadgets. These strings are each 1 byte long.

This menu option is used when you want to use some text or word processor to format and print a Thinker document.

Either the entire Thinker document is exported (the "Project" sub-menu option) or a single branch (the "Branch" sub-menu option) is exported. If only a single branch is exported, it is the anchor branch (the branch at the top of the window).

Import Text

This option asks Thinker to read a flat ASCII text file and to incorporate it into the active Thinker document immediately after the anchor statement (the statement at the top of the window). Thinker first puts up a file requester to allow the choice of a file. Then the Import Selection requester is put up for final setup. The "Level" gadget in the requester selects whether the text is added at the same level as the anchor statement or at a level down from the anchor statement. Selecting the "SAME" or "DOWN" gadget toggles from one state to the other.

If "1 LF" is chosen, Each line in the ASCII file becomes a

statement in the **Thinker** document.

If "2 LF" is chosen, Each paragraph of the ASCII file becomes a statement in the **Thinker** document. A paragraph is determined by the appearance of a blank line (one containing no characters other than a Line Feed (Unix record delimiter)) separating lines of text. The indentation of the text is used to construct the hierarchy appropriate to **Thinker** documents. Paragraphs that begin with labels have the labels added to the **Thinker** document.

The style gadgets are the same as in the Export requester. The appearance of these characters in the ASCII text will cause **Thinker** to set the corresponding style for the text between the characters. There is no way to import color information.

Import Text is the dual of Export Text. If you export a **Thinker** document with the "Indent Prefix" equal to the "!" character, you get a flat ASCII text file that looks exactly like the document appears in the **Thinker** window. If you Import this file using the "2 LF" option, the resultant **Thinker** document will be identical to the original except for the loss of color information.

To create a **Thinker** document from an ASCII text file, first use the "New" menu option to create a new Project. Then use the "Import Text" menu option to import the text file at the "SAME" level. Delete the copyright notice (first statement) and you have a **Thinker** document.

Imported statements are truncated at 2000 characters.

Get Options

The option settings, behavior settings, FKey values, and Font selection are read from the icon of the document displayed in the window. If no icon exists, the **Thinker** default settings are selected. The window is redisplayed to reflect any changes in the options settings.

Save Options

The option settings (from the Options menu), FKey values, and Font selection, are recorded in the icon of the document displayed in the window. An Icon will be created if none exists. These options become the default options when the document icon is opened from the workbench.

In addition, a complete environment description is stored in the Icon of the file displayed in the active window. If several windows are open on the screen when Save Options is selected then a description of all the open windows is saved in the Icon of the file displayed in the active window. When the Icon is later double clicked to launch **Thinker** all the windows will be opened with their various option settings and the files displayed at the time of the Save Options selection are displayed. Any window that was positioned over a labeled statement at the time of the Save Options selection will be positioned at the same labeled statement when **Thinker** is launched. Otherwise the window will be positioned over the origin of the document.

If several files are displayed, a different set of options can be saved in each file icon. Change the options and then activate a different window and use Save Options to save the new options in the file icon of the newly active window.

NOTE: Not all of the options are saved in the descriptions of the secondary windows. Only Clip, Spacing, and Number of lines are saved for windows other than the primary window. The primary window is the window that matches the active window when the Save Options menu selection was made.

Build Index

A branch with the label "index" is added after the anchor statement. There is a statement within this branch for each label that appears in the **Thinker** document. The statements are in alphabetical order and the labels are surrounded by <> characters.

The Index branch can be edited by adding comments or left as is

for use as see-thru links to the rest of the document.

Quit - RAMiga Q

Thinker terminates after closing all windows. The user may be asked to respond to "Discard Modifications" prompts for any project that has been changed and for which no "Save" has been done.

There are two rounds of prompts when Quit is selected with outstanding updates. The first prompt for each file with outstanding updates gives the user the option of cancelling the Quit selection. The second prompt is the last chance to save updates.

Edit

The Edit menu is used to perform normal word processing functions on the **Thinker** document.

Cut - RAMiga C

If the cursor is swept over text while holding down the left mouse button, text is selected. The Cut menu item selection removes the selected text and places it on a **Thinker** internal Clipboard.

Copy - RAMiga V

Selected text is copied onto the **Thinker** internal Clipboard as in Cut, but it is not removed from the statement where it was selected.

Paste - RAMiga P

The text from the **Thinker** internal Clipboard is copied into the statement at the cursor. If the cursor is not within a statement, no action is taken.

Search

Selecting the Search menu item causes **Thinker** to put up a Search/Replace requester. The user may enter a search string and a replacement string. The search is performed without regard to the case (upper or lower) of the search string. The replacement string is inserted exactly as typed except that the case of the first character is made the same as the case of the first character of the located string. The "Replace as Typed" behavior will eliminate the first character case switching.

The search is activated either by typing a <Return> at the end of the Search String OR by selecting the Find gadget. A replace is activated either by typing a <Return> at the end of the Replace String OR by selecting the Replace gadget. If the Replace String is NULL the search string is deleted when it is located in the text.

Search and Replace treat the **Thinker** document as a linear document starting with the anchor statement and continuing until the end of the document.

Search/Replace speed is improved if the window height is reduced.

Next - RAMiga N

The previous Search/Replace function is repeated beginning at the next character in the document.

Stop

Selecting the Stop menu item will terminate an long running **Thinker** operations. "Stop" will terminate "Search", "Spell", "Import", "Export", Print, and "Build Index" functions. You may have to select "Stop" several times for some operations.

Clipboard

The first part of the **Thinker** clipboard is displayed in its own small window. This window may be closed by selecting its close gadget. Data on the clipboard can not be modified and is not affected by closing the window.

Edit FKeys

Note: ARexx must be installed for FKeys to work.

The 10 Function Keys (F1 through F10) can each be programmed to issue a **Thinker** command up to 64 characters long. The Edit FKeys menu selection brings up a requester with 10 string gadgets for the 10 Function keys. Selecting OK updates the current definitions of the Function Keys and selecting CANCEL leaves the definitions unaltered. The Function Key definitions are saved permanently using the "Save Options" menu selection.

The Function Key strings are sent to **Thinker's** own ARexx port as an ARexx command as if the command had come from some ARexx program. The command can be any of the ARexx commands described in the section on the **Command Port of the ARexx Interface**. **Thinker's** ARexx interface is designed so that any ARexx command that it receives that is not recognized as a valid command is treated as a macro and is submitted to the ARexx Master Task as a macro invocation. The section **Macros of the ARexx Interface** contains a more detailed discussion of the macro combination of **Thinker** and ARexx.

Another way of looking at the Function Keys is as an equivalent Jump Link command. Since Hypertext links have been extended to ARexx ports, a Jump Link to an ARexx port results in a message being sent to that port (the label portion of the link). A function key programmed with the string "search first gibberish" becomes a Jump Link command to the link <Thinker,search first gibberish> which sends the "search first gibberish" command to **Thinker's** ARexx port "Thinker".

Another example is a Function Key programmed with the string "advance". Selecting this function key sends the command "advance" to **Thinker's** ARexx port where it is treated as a macro (not a **Thinker** command) and is sent to the ARexx Master Task. The ARexx program "advance.thnkr" is run and sends **Thinker** a series of commands that result in the cursor being advanced to the beginning of the next word in the statement being edited.

Commands

All of the **Thinker** commands are available from the Command Menu. Most of the commands (except for the uncommon Jump commands) are also available by selecting the on-screen gadgets. Many of the command menu selections have Command Key equivalents that allow for quick selection without reaching for the mouse. Most commands will require that the mouse be used to select the target of the command anyway so little is gained in most cases by using the Command Key.

Jump

All of the Jump commands end with a confirmation requester. This requester gives the option of performing the Jump within the current window, opening a new window, or completing the jump in another **Thinker** window.

When the "Other Window" option is chosen **Thinker** gives the user the opportunity to select a different window to complete the jump. Move the pointer into the **Thinker** window where you want to complete the jump, click the left mouse button and select the desired option from the jump confirmation requester.

Mark - RAMiga M

The pointer is changed to a hand. Position the hand so that the finger points to the statement that is to become the anchor statement of the window and select the statement by clicking the left mouse button. The "Qjmp" gadget has the same function.

Link - RAMiga L

The "Link Address" requester is brought up. If the desired link is visible on the screen (either a word or a string delimited by "<" and ">") select "Mark" and position the hand so that the finger is pointing to the link and select the link with the left mouse button.

If the link is not visible on the screen, type the link (without the

"<" and ">" characters) in the string gadget and type the <carriage return> key. Select "Previous" to recall the last value typed into the string gadget.

Select the File Requester gadget to bring up a file requester to scan for **Thinker** documents or workbench applications. See the Section titled "File Requester Explained" for an explanation of how to use this requester.

Return - RAmiga R

The 6 previous locations in the document are summarized in a requester. Select the designated statement from the selection list in the requester with the mouse.

Forward - RAmiga F

There is a Jump to the last statement on the screen. Jump Forward can be used repeatedly to scroll forward in a **Thinker** document.

Up - RAmiga H

The pointer is changed into a hand. Move the hand so that the finger selects a statement. The Jump is relative to the selected statement and the window is moved to the statement one level closer to level 1 in the hierarchy.

Down - RAmiga T

This Jump is like Jump Up except that the window is moved to the next statement one level deeper in the hierarchy. This is more likely to be useful if the resultant Jump is to a level higher than the current clipping level (toward more details).

Preceding - RAmiga J

The Jump is to the statement that precedes the marked statement at the same level. Note that if the marked statement is the first statement in a group that is subordinate to some statement, the Jump does not complete as there is no statement

preceding the marked one at the same level.

Succeeding - RAmiga K

The Jump is to the statement that follows the marked statement at the same level. If the statement marked is the last statement of a group subordinate to some statement, the Jump does not complete as there is no statement following the marked one at the same level.

Next - RAmiga Y

The Jump is to the statement that follows the designated statement regardless of level. Jump Next is like a scroll forward one statement.

Back - RAmiga W

The Jump is to the statement that precedes the designated statement regardless of the level. Jump Back is like a scroll backward one statement.

Origin - RAmiga O

The Jump is to the origin statement of the document.

Link Cursor - RAmiga E

Like Jump Link (Mark) except that the link is found under the cursor rather than being pointed to with the mouse. This allows the keyboard user to move the cursor over a link and then use Jump Link Cursor to do the Jump Link. Double clicking the left mouse button performs the same function.

End

Jump to the last statement in the document.

Delete

Either of the sub-menu items (branch or group) cause portions of the **Thinker** document to be removed. The statement to be deleted need not be in the active window. If you want to delete a statement and leave the subordinate statements, you must move the subordinate statements before doing the delete.

Branch

The "Address" requester is brought up and the user may select "Mark" or type the label of the branch to be deleted in the string gadget of the requester. If "Mark" is selected then the pointer is changed to a hand and the user moves the hand over the selected branch and selects the branch with the left mouse button. If the user selects "confirm" from the "confirmation requester" the entire branch is deleted.

Selecting "Previous" recalls the previous value typed into the string gadget.

Group

The "Begin Address" and "End Address" requesters allow the user to select the range of the group. Note: if "Mark" is used to select the begin and end of the group, both ends of the group must be marked in the active window.

Selecting "Previous" recalls the previous value typed into the string gadget.

Plex

The "Plex Address" requester is brought up. The user selects any statement in the plex.

Selecting "Previous" recalls the previous value typed into the string gadget.

Insert

The Insert After, Insert Down, and Insert Up commands are all executed immediately without any further interaction. The new statement is inserted relative to the statement with the active typing cursor displayed (the vertical red line that appears when the left mouse button is pressed with the pointer positioned over the text). No action is taken if the typing cursor is not visible. Insert Mark is equivalent to the Insert gadget at the top of the window.

If the inserted statement is near the bottom of the screen or off the screen, **Thinker** does an automatic jump to the statement preceding the inserted one. This automatic jump is disabled using the Options menu.

Insert Mark

The pointer is changed into a hand. Move the hand so that the finger is over a statement. Select the statement with the left mouse button and a confirmation requester is brought up. The confirmation requester offers options as to where relative to the selected statement the new statement is to go.

After

The new statement is inserted immediately after the selected statement at the same level.

Down

The new statement is made the first subordinate statement to the selected statement. If there are already subordinate statements, the new statement is put in front of the existing subordinate statements.

Up

The new statement is inserted directly above the selected statement. In fact, the Plex of which the selected statement is a part is split into two and the second half of the Plex is made subordinate to the inserted statement.

Insert After - RAmiga I

An immediate version of Insert Mark where a statement is inserted after the statement in which the cursor is located. No confirmation is required. This is for power input from the keyboard.

Insert Down - RAmiga D

An immediate version of Insert Mark where a statement is inserted down from the statement in which the cursor is located. No confirmation is required. This is for power input from the keyboard.

Insert Up - RAmiga U

An immediate version of Insert Mark where the statement is inserted up from the statement in which the cursor is located. No confirmation is required. This is for power input from the keyboard.

Insert Image

The image in the most recently selected "Image Window" is inserted into the document as a statement (The Image Window should be at the Front of the display for proper operation). These image statements may not have any labels and contain no text. If several image windows are open then the one that has most recently been opened or most recently touched is the one whose image is inserted. To select the image to be inserted, simply put the cursor over the image window and click the left mouse button.

If you change the Workbench palette after inserting an image into a document, the image colors may become distorted. The conversion of image colors to match the Workbench happens only when the image is first displayed in the Image Window.

Any of the 3 image displays may be inserted into a document. Please note that a WB or WBX window takes 8000 bytes of file space on a non-interlaced screen and 16000 bytes on an

interlaced screen. An Xfer window takes only 2000 or 4000 bytes.

Copy

The Copy command duplicates portions of a **Thinker** document and inserts the copy at the location requested. The copied portions may be inserted at any level in the document. Copy may be to a different **Thinker** document. Note that there need not be a window open displaying the destination document. If the user types the name of a labeled branch in another document as the destination of the copy, the copied portions of the document will be placed in the destination project.

The user may not copy a Branch, Group, or Plex to anywhere inside of the Branch, Group, or Plex.

Statement

Copy Statement only copies the statement and not any subordinate statements as in Copy Branch.

The "Address" requester is brought up. The user may select "Mark" and move the hand to point to the statement to be copied or type in the label name in this document or in another document. Select "Previous" to recall the previously typed label name.

After selecting the source of the copy, the "Target requester is brought up and the user makes the same choices as in the "Address" requester.

Finally the "confirmation" requester is brought up and the user selects the level relative to the target statement for the copied statement. As in Insert Up, designating Up splits the plex at the designated statement and makes the second part of the plex subordinate to the last statement of a group of copied statements.

Branch

As in Statement but the entire branch is copied.

Group

The "Begin Address" and "End Address" requesters allow the user to select the range of the group. Note: if "Mark" is used to select the begin and end of the group, both ends of the group must be marked in the active window.

Plex

The "Plex Address" requester is brought up. The user selects any statement in the plex.

Selecting "Previous" recalls the previous value typed into the string gadget.

Move

Move is like a Copy with the source of the copy deleted after the copy. In fact that is how it is implemented when the move crosses Thinker documents.

Branch - RAmiga B

As in Copy.

Group - RAmiga G

As in Copy.

Plex

As in Copy.

Sort

The Sort command provides for re-arranging a selected group of branches according to the alphabetic sequence of the statement label or its contents. The options are: File, Branch, Group, and Plex.

In the case of File all first level branches are sorted. The Sort Requester is put up immediately to begin the sort process.

In the case of Branch, Group, and Plex, the appropriate "range" requesters are brought up as in "Delete", "Copy" or "Move". After the range has been selected, the Sort Requester is put up to begin the sort process.

See the section titled "Sort Requester Explained" for a description of the sort requester.

Options

Options are set on a window by window basis. The default option settings for a new window are those saved in the project icon (.INFO file) which was used to launch **Thinker** not the project icon of the "jumped to" project. You can set the default options for a particular project any time the project is displayed in a window. Save Options always updates the icon of the displayed project (name shown in the window title bar) with the option settings of the window.

It is possible to set the Clip level, Spacing, and Number of lines in a single menu operation. While holding the right mouse button down, move the mouse to expose the various menu selections and select the set of options with the left mouse button. When the right button is release, the window will be redrawn with the new options active.

Clip

The clipping level may be set to 1, 2, 5, or 10 using the sub-menu item selections. This sets the number of levels of the hierarchy that are displayed on the screen.

Clip 2 - RAmiga 2

Clip 5 - RAmiga 5

Clip 10

Spacing

The number of blank lines between statements is set to either 0, 1, or 2 using the sub-menu item selections.

Spacing 0 - RAmiga 0

Spacing 1 - RAmiga 1

Spacing 2

No Jump Confirm

Normally the Jump command puts up a final confirmation requester that allows the user to open a new window or have the Jump take place in another Thinker window. If this option is activated, the Jump completes in the active window without any confirmation. By using Jump Forward and Jump Return without confirmation, the user can page forward and backward in a document.

No Jump Confirm interacts with the Select Jump Target behavior. When Select Jump Target is active, only those windows with No Jump Confirm selected will activate the Automatic jump in the selected window.

Show Date

When Show Date is selected each statement is followed by a line that displays the date and time that the statement was last modified. The date is adjusted to the right hand margin so this option is best combined with Spacing of 0.

Statement Lines

This option controls the number of lines of each statement that is displayed on the screen. The sub-menu item selections are for 1, 2, 5, or 99. If 1 is selected only the first line of each statement is displayed and the document looks like an outline.

Lines 1 - RAMiga X
Lines 2
Lines 5
Lines 99 - RAMiga Z

Black on White

The text is displayed in black characters on a white background.

Auto Jump on INS

With this option selected and a statement is inserted near the bottom of the screen, **Thinker** automatically positions the window at the statement back 1 from the newly inserted statement.

Read Cache On

If you do not have a Floppy Accelerator like Facc II installed, you will probably want to use this option. With this option selected, a reasonable number of statements that have been read from the disk are saved in memory. This buffer is cleared when "Discard Modifications" or "Apply Modifications" is selected. If you have more than a few projects open and you are using the Read Cache On option, quite a bit of memory is used to store the previously read statements. Facc II does a much better job.

The default for Read Cache On is selected. This change was made as a result of press reports of slow behavior when refreshing the screen. If you do have FaccII **BE SURE TO TURN OFF THE READ CACHE**. It is a long explanation but with both caches on the behavior can be the same as neither cache on.

Flag Forced

When statements are displayed as a result of using the statement "+" gadget, you are seeing statements that are below the global clipping level. If you wish to have these statements flagged so that you can be aware of this behavior then select this option.

Hide Labels

With this option selected **Thinker** will not display the labels on the screen nor will it print them or put them into an ASCII text file. When **Thinker** documents are printed or put into ASCII text form it is unlikely that labels will do more than clutter up the appearance of the text. This option allows the user to use labels for an effective online document and still print the text in an uncluttered form.

Error message box

With this option selected a box is drawn in the upper right corner of the window and any error messages will be displayed in this box until some activity indicates that the error message has been seen. Typing a key, activating a gadget, or selecting a menu will clear the error message box. If the option is not selected, **Thinker** will "Beep" the screen when an error is detected.

Enable See-Thru

A statement that consists solely of a link surrounded by < > can be treated as a "See Through Link". The first character of the statement must be a "<" and the last character of the statement must be a ">". When a "See Through Link" appears on the screen and the Enable See-Thru option is enabled, the "See Through Link" is replaced by the statement that is named in the link.

The statement is marked on the screen with a ">" character in the left margin of the window in order to flag these statements. You may move or copy these statements as you would any statement. What is happening, however, is that the "See Through Link" is copied or moved rather than the displayed statement. You can observe this after copying or moving the statement and then disabling "See Through" and observe that the "See Through Link" was actually moved or copied. What happens when you edit the displayed statement depends on the setting of the "Make local copy" option.

"See Through Links" allow portions of one document to appear as if they were part of a second document. The information is stored

only once in the original document but is viewed as if it were part of other documents via the "See Through Link". This feature allows the creation of new documents that are re-organizations and re-groupings of previously created documents. "See Through Links" are not recursive because a statement with a "See Through Link" may not be labeled.

When you use links displayed in a "See Through" statement, the links are resolved in the document from which the statement was extracted (unless the link contains an explicit document name as part of the link). Thus, if an extracted statement excites interest, double clicking on any link displayed in the extracted statement may carry you into the document from which the statement was extracted.

Make local copy

When you modify the contents of a statement that is displayed as the result of a "See Through Link" you either change the original document or copy the statement into the displayed document replacing the "See Through Link".

With "Make local copy" enabled the data from the original document is copied into the displayed document replacing the "See Through Link". You will notice that the ">" character in the left margin disappears indicating that there is no longer a "See Through Link". This option allows a new document to contain a variation of the original document. However, changes in the original will no longer be reflected in the new document because of the copy operation.

With "Make local copy" disabled, the change is written into the original document and the displayed document is not altered. When you close the last **Thinker** window, you will be greeted with requesters asking if you want to apply modifications made via the alteration of statements displayed because of a "See Through Link". If you want to apply or discard these modifications before leaving **Thinker** you must Jump Link to the altered document and use the Project menu options there.

Once you start changing a "See Through" statement you cannot

change your mind with regard to local copy. You must select the correct option, Enable or Disable, before you begin editing.

Allow Dup Labels

The default is to allow more than one statement to have the same label. In cases when this is not desired, enabling this menu option will disallow duplicate labels. When duplicate labels are disallowed, any label that duplicates an existing label is removed from the text. "(options,menus)..." becomes "(options,)..." if the label "menus" already exists in the document.

Spell

The Spell feature of **Thinker** makes use of an external lexicon file. This file is 128K bytes and is read into memory when the Spell feature is Initialized. The dictionary is only 33,000 words so there will be many correctly spelled words detected in your documents that will not be in the dictionary. These words can be added to the dictionary and the new dictionary saved for later use.

The dictionary is actually a collection of hashes of the words and the probability of identifying an incorrectly spelled word as correctly spelled is about 1/30000. As more words are added to this dictionary the probability of such an identification increases. The dictionary should not contain many more than 40,000 words.

Initialize

Select this option to read in the dictionary file. The file is looked for in 3 places. First, the Drawer containing the **Thinker** document which was double clicked is searched (or the working directory if started from the CLI) for the file "thinker.lex". Second, the Drawer in which the **Thinker** program was found is searched for the file "thinker.lex". Third, the file "LEXDIR:thinker.lex" where LEXDIR is a tool type in the **Thinker** icon. A ToolType of LEXDIR=THINKER2: will cause **Thinker** to look for the file "THINKER2:thinker.lex" as a last resort. The default value for LEXDIR is "Thinker2".

128K of memory is set aside to read in the dictionary and the

Spell option is enabled.

Spell

The spelling check begins at the cursor and continues to the end of the document. Each word is isolated from the text and checked against the dictionary. A word is a collection of non-punctuation characters between punctuation characters. Contractions will be treated as two words. If the word is not located in the dictionary a requester with 5 options is brought up.

Accept

The word is added to the dictionary. Future occurrences of the word in the text will be recognized as correctly spelled.

Stop

The spelling check is terminated. The Edit menu, Stop sub-menu item can also be used to stop the spelling checker when the requester is not up.

String Gadget

The isolated word is displayed in the string gadget. If the user wants to correct the spelling of the word, it is done in the string gadget.

Replace

The word in the string gadget (presumably after being corrected) is put into the text in place of the isolated word.

Continue

The word is skipped and the spell check continues.

Save

The user can save an updated dictionary after the spell check completes. Selecting this option overwrites the dictionary on disk

with the updated dictionary in memory. The exact file that was read when the Initialize menu option was selected is overwritten.

Merge

This is a way of building new dictionaries. Each word in the text is assumed to be correctly spelled and is added to the dictionary. It is as if the user "Accepted" every word.

A fresh new dictionary can be built using the following technique.

Choose the Initialize function without having the **thinker.lex** file available. Respond "Cancel" to the message requesting the THINKER disk. The Spell options are enabled with an empty dictionary.

Jump Link to a document that contains the words that constitute your new dictionary.

Use the Merge option to add these words to the **Thinker** dictionary.

Use the Save option to save the "**thinker.lex**" file in the Drawer with the last launched document or **Thinker** itself.

Free

The memory used to read the dictionary file is freed.

Behaviors

The following behavior selections affect all active windows. These selections tailor the behavior of **Thinker** for particular uses. Single Click Jump, as an example, is wonderful for browsing a document but makes editing a document difficult. As with the Options settings, the default behavior settings can be edited into the **Thinker** icon using the INFO program. Behavior settings in launched projects will override these defaults and the Get Options menu selection may be used to select Options and Behaviors from the icon of the active file.

Single Click Jump

With this behavior selected only a single mouse click on a link is required to execute a Jump Link. This behavior is best used when browsing a document rather than for editing it. It is still possible to edit while this behavior is selected but you will have to cancel jumps if you try edit a link.

Allow `_[];#{}|`

Punctuation characters are normally stripped from labels and links when doing the search. When this behavior is selected, the selected punctuation characters are kept as part of the label and link. Be aware that if you add some labels and links with this behavior selected and some without you may become confused as there is no behavior that makes both types of labels reachable at the same time.

Note also that most of the ALT characters are not, and never have been treated as punctuation. All of these characters are kept in labels and links.

Typeover Disable

When SEARCH is used to locate text, the text is left selected. Should you type at this point, the selected text is replaced by your typing. Selecting this behavior disables the automatic removal of the selected text. This behavior applies to selected text no matter how it was selected. If you select text in order to change its style, this behavior will eliminate the accidental removal of the selected text should you type at the keyboard before de-selecting the text.

Show Anchor Level

When this behavior is selected, the level number of the anchor statement is displayed to the left of the statement.

Reverse Video Cursor

A block cursor is used in place of the single vertical line. This block cursor is easier to see on low resolution monitors.

Select Display Font

Thinker will normally use the default Workbench font to display a document. The default Workbench font can be altered to a font other than TOPAZ 8 and **Thinker** will adjust its display accordingly. However, it should be noted that the rendering of menus with Workbench fonts higher than 10 pixels does not seem to work correctly.

The "Display Font" menu selection can be used to select different fonts for the document display without changing the font used for the menus, gadgets, and requesters.

The sub-menu option "Select" brings up the **Thinker** file requester on the FONTS: device (directory). To choose a font, follow the directions in "File Requester Explained" to select the desired font and size. An example is "FONTS:DIAMOND/12" for the Diamond 12 pixel high font. The sub-menu option "Workbench" resets the display font to the Workbench font.

Select Jump Target

One of the ways to use Hypertext efficiently is to have more than one window open at a time. In this way the main portion of the document can be displayed in one window while other windows are used to follow links. Selecting this behavior simplifies the jump sequence by eliminating the need to select "Other Window" and then "This Window" when doing jumps in another window.

One of the open windows can be selected as the window in which to do ALL jumps. Choose the "Select" sub item from the "Select Jump Target" item and move the hand to point to the designated window. Press the left mouse button and the selected window will be drawn with a black border instead of the usual white border. All subsequent jumps will be performed in this window. When you click on a link the designated window will move to display the labelled statement.

This behavior is disabled with the "Clear" sub item of the "Select Jump Target" menu item. The behavior is also terminated when the designated window is closed.

There is an interaction of "Select Jump Target" and "No Jump Confirm". Only windows with "No Jump Confirm" selected will take advantage of the automatic use of the selected Jump Target window.

Default Link File

An interesting use of Hypertext is to have a "dictionary" file to accompany a file. This dictionary file has labels for words and phrases in the main file and provides detailed related information. However, links in the main file would normally have to have the dictionary file name as part of the link and you could not use single words in the main document to link to the dictionary document.

By selecting a file to be the "Default Link File" you can have the desired behavior. With the dictionary file selected as the "Default Link File" all links (without explicit file name portions) will be resolved in the dictionary file.

If, and only if, you have previously selected a "Default Jump Target" window, you can designate the file in that window to be the "Default Link File" by selecting the "Select" sub item of the "Default Link File" menu item. Any window that is displaying the default link file will be drawn with a red border.

Example: Use Jump Link to open a window over the dictionary file. Make this window the "Default Jump Target". Choose the "Select" sub item of the "Default Link File".

The default link file is cleared by selecting the "Clear" sub item of the "Default Link File" item or by selecting the Close item in the Project menu in the window displaying the default link file and it is not displayed in any other window.

Replace as Typed

This behavior disables the case mapping of the first character of replacement strings (in Search/Replace). With this behavior selected the replacement string is not altered when it is inserted into the document.

Close Unneeded Files

With this behavior selected, files are closed when they are no longer displayed in any window. You will be asked to insert the disk that contains the file when the file is closed. However, you will not be asked to insert the disk when "Quit" is selected.

The normal sequence of a Jump Link is to validate the link before closing the file being displayed. This means that at one point during the Jump both files are open. If the files are on separate disks and you are not careful about which disk you remove (or if you have only one drive) you will have to flip-flop disks during a jump to a new file with this option selected.

However, If you have selected a Jump Target window the sequence is altered. The old file is closed and the Jump Target window is blanked before opening the new file. This sequence will not require any floppy disk flip-flops but will leave the Jump Target window blank if the link is not valid.

Enable Dot Commands

Links have an extension separated from the label with a ".". The extension describes a series of commands that are acted on after the label is located. Dot Commands may be combined. A link with Dot Commands looks like <Dot Commands.dt> These commands are described below. The Link <Dot Commands.dtfhighlight> directs **Thinker** to jump to the statement labeled "Dot Commands", jump Down, jump to the Tail of the Plex, and then Find the word "highlight" and highlight it. NOTE: If you have labels with "."s in them, the dots are ignored unless you select this "Enable Dot Commands" menu selection and then they are interpreted as introducing a Dot Command sequence.

D - Down. The link *xx.d* directs **Thinker** to jump to the label **XX** and then Jump Down to the first subordinate statement.

S - Succeeding. The link *xx.s* directs **Thinker** to jump to the label **XX** and then Jump Succeeding to the next statement at the same level.

P - Preceding. The link *xx.p* directs **Thinker** to jump to the label **XX** and then Jump Preceding to the previous statement at the same level.

T - Tail. The link *xx.t* directs **Thinker** to jump to the label **XX** and then Jump to the last statement at the same level (The Tail of the Plex). This command is usually combined with *d* in *xx.dt* which means to jump to the end of the plex subordinate to the statement labelled *xx*.

F - Find. The link *xx.fstring* directs **Thinker** to jump to the label **XX** and then find *string* in the statement and highlight it. This represents a Link to a word or phrase.

Style Delimit Links

When enabled, this behavior makes it possible to eliminate the use of "<>" characters to delimit links with embedded blanks. Any sequence of non-plain text (bold, italic, underline, or some non-plain color) will be treated as a link delimiter.

The link <This is a link> and *This is a link* are equivalent.

If a non-plain string contains "<>" characters, the "<>" characters have priority in delimiting the link.

The link This <is a> link and *This <is a> link* are equivalent with the link being "is a" in both cases.

The style of the characters is merely used as an easier to read substitute for <>. **Thinker** does not automatically change the style of links or labels. Links and labels are still just matching text strings. Styles are simply another way of delimiting the links in the text.

Note that the addition of styles to a document can increase its size significantly. Documents with very short statements will double in size if each statement has at least 1 non-plain character.

Style

The Style menu is used to select the style of text that is typed from the keyboard or to change the style of selected text. Changing the style of **selected** text does not affect the style of text later typed at the keyboard. In addition, if "Plain" is the current type style, characters are typed in the style of the character that immediately precedes it.

Note: Pen colors are absolute and are not affected by the "Black on White" Option which reverses the Plain and Background colors.

Plain

Makes the selected text plain (no colors or styles) or makes the style and color of future text equal to the style and color of the immediately preceding character (if no text is selected).

Bold - Makes selected text **bold**.

Italic - Makes selected text *italic*.

Underline - Makes selected text underlined.

FG Pen1 - Selects color 1 for the foreground pen.

FG Pen2 - Selects color 2 for the foreground pen.

FG Pen3 - Selects color 3 for the foreground pen.

BG Pen1 - Selects color 1 for the background pen.

BG Pen2 - Selects color 2 for the background pen.

BG Pen3 - Selects color 3 for the background pen.

Make UPPER - The selected text is forced to UPPER case.

This has no effect on future typing.

Make lower - The selected text is forced to lower case.

This has no effect on future typing.

Memory

This menu shows the amount of memory available to Thinker.

Gadgets Explained

Most of the common commands can be selected with on-screen gadgets. Gadgets have the advantage over menus as they are easier and faster because a single click gets you started without having to scan the menu selection. However, you will be presented an extra requester in some cases where sub-menu items would be used in the menu selection.

The gadget box in the upper left corner of the screen displays a clock which is updated every few seconds. This clock only runs when the window is active and catches up quickly when the window is activated after a long inactive period.

Cancel

If the pointer is in the form of a hand while **Thinker** is waiting for the selection of a statement, the Cancel gadget stops the command.

Insert

The pointer is changed into a hand. Move the pointer to select a place for the new statement. The new statement will go after the selected statement. Select the statement with the left mouse button and a confirmation requester is brought up. Select the option for the relative location of the new statement.

Delete

The "Delete" requester is brought up. Select the type of delete (Branch or Group) and the command proceeds as in the Delete Branch or Delete Group menu selection.

Jump

The "Jump" requester is brought up. This requester allows selection of the common Jump commands (Succeeding, Preceding, Link, Mark, Up and Return). After one of the options is selected, the Jump command continues as in the Jump Menu.

Clip +

The Clipping level is increased by 1. One more level of the hierarchy will be visible for the entire document.

Clip -

The Clipping level is decreased by 1. One less level of the hierarchy will be visible for the entire document.

Copy

The "Copy" requester is brought up. This requester allows selection of the Copy commands (Statement, Branch, Group) that are available in the Copy menu. After selection of one of the options, the command continues as in the Copy menu.

Move

The "Move" requester is brought up. This requester allows selection of the Move commands (Branch, Group) that are available in the Move menu. After selection of one of the options, the command continues as in the Move menu.

Qjmp

Qjmp is a quick way to Jump to a statement on the screen. Qjmp is equivalent to a Jump Mark with automatic confirmation (same window).

Zoom Gadget

Selecting the zoom gadget (left of the Front/Back gadget in the drag bar) toggles the window between a full screen size and the last size it had that was smaller than a full screen. If you size a window to fit in the lower right corner of the screen, you can expand the window to full screen size by selecting the zoom gadget. Selecting the zoom gadget a second time will shrink the window back to the lower right corner of the screen.

Scroll backward

The scroll gadget in the top right-hand corner of the window moves the display one statement toward the origin of the document each time it is selected. If you "hold down" the gadget, scrolling continues until the origin is reached.

Scroll forward

There are two Scroll forward gadgets in the lower right-hand corner of the window. The gadget with the single arrowhead moves the display one line toward the end of the document each time it is selected. If you "hold down" the gadget, scrolling continues until the end is reached.

The second Scroll forward gadget has a double arrowhead and scrolls forward by statements rather than lines.

Statement +

The appearance of this gadget in front of a statement indicates that the global clipping level is such that there are subordinate statements that are not displayed. Selecting this gadget will cause the next level of statements to be displayed for this branch only.

Statement -

The appearance of this gadget in front of a statement indicates that the statements that are displayed subordinate to this statement are displayed as a result of selecting the statement "+" gadget. Selecting this gadget suppresses the display of these lower statements.

Jump Forward

A third scrolling gadget appears in the lower right border. This gadget is an inverse color arrow. Selecting this gadget is exactly like doing a Jump Forward command with no confirmation required. The window is scrolled forward so that the last statement on the screen becomes the anchor statement. If a statement does not fit in a screen, Jump Forward scrolls so that the next portion of the statement is displayed.

Jump Return

A second scrolling gadget appears in the upper right border. This gadget is an inverse color arrow. Selecting this gadget is exactly like doing a Jump Return command with no confirmation required. The Jump Forward and Jump Return gadgets can be used to quickly scan a document.

Warnings Explained

There are eight Warning Requesters that indicate that some action is required by the use to avoid some undesirable results.

Are you sure?

"Revert" has been selected. You must confirm this action to avoid unintentional loss of work.

Discard Changes

An attempt to exit **Thinker** with some unapplied modifications to one or more documents. A requester will be presented for each document with unapplied modifications. The "DISCARD" option will cause the modifications to be discarded. The "SAVE" option will cause the modifications to be applied to the document.

If there were several documents with unapplied modifications, the requester will be repeated for each document.

Memory Low

This warning does not require immediate action. **Thinker** has detected a low memory situation. Memory can be freed by selecting "Save" or "Revert" from the "Project" menu or by selecting the "Free" option of the Spell menu.

If no action is taken and **Thinker** actually runs out of memory, you may get a visit from the GURU. The most likely happenstance is that **Thinker** will simply close all windows and exit. **Thinker** documents are rarely ruined by visits from the GURU as all the modifications are in the memory of **Thinker**. Work is lost!

Disk Write Error

The disk may be full or write protected. AmigaDOS will probably have given one of its own Alerts prior to the appearance of the **Thinker** warning requester. There is no action required (other than acknowledging this requester). The modifications to the **Thinker**

document are still in the memory of **Thinker**. However, if the disk became full while writing the updates, it is necessary to take some action to avoid a ruined **Thinker** document.

Use a CLI (PopCLI if you are running **Thinker** from the CLI) to delete some projects on the disk and then retry "Apply Modifications".

Memory low - If IMPORT continues

A memory low condition was detected while importing a text file. **Thinker** needs to save the changes made to the file in order to continue with the Import. If you select Continue, **Thinker** will do a "Save" operation and continue. If you select "Stop" the Import is terminated. You may "Revert" to the previous version.

Memory low - Export Aborted

A memory low condition was detected while exporting a **Thinker** document to a text file. This condition arises when the "Read Cache" of **Thinker** is enabled. Disable the Read Cache and try again.

Memory low - Build Index Aborted

A memory low condition was detected while building an index. Build Index may add a lot of statements to a **Thinker** document. These statements are all in the Write Buffer until the "Save" menu option is selected. If you have a small machine and a large document, you may not be able to use Build Index.

Thinker is Lost....

If more than one window is open over a single document and you use the "Revert" menu selection to back out some changes, it is possible that some windows may display this message. This situation arises because the anchor statement of the window with the message was one of the statements that was removed by the "Revert" request.

Either close the window or do some Jump command in the window to select an anchor statement that exists.

File Requester Explained

The **Thinker** File Requester consists of two file selection areas and several gadgets. File names from the Drawer are displayed in the "file name display" area or a file name may be typed into the "file name string gadget". The scrolling gadgets on the right side of the "file name display" area move the display over the entire file list.

When the file requester is first brought up **Thinker** begins to scan the Drawer named in the initial contents of the "file name string gadget". This Drawer is either the one that contains the **Thinker** document if a Project Icon was selected, the one that contains the **Thinker** application if the Tool Icon was selected, or the Drawer named in the string gadget of the "Link Address" requester. This string gadget may have been filled in by typing or selecting the "Previous" gadget in the "Link Address" requester before selecting the File Requester gadget. The scan of the Drawer is stopped by selecting a file in the file name display area or selecting one of the gadgets (Parent, OK, or NO).

Assuming that the display is static because the scan of the Drawer is complete, the following actions may be taken.

OK

The file name displayed in the string gadget is used as the target of the Jump Link. If there is no comma (there is no label name after the file name), one is added to complete the link as a link to a file name. The file name selected may be any valid link including applications and projects. If there is a comma followed by a label name, the jump is completed to the labeled statement in the target document (The label is ignored if the file is not a **Thinker** document).

NO

The requester AND the Jump Link command is canceled.

Select name from display area

The name selected is added to the file name in the string gadget. If the contents of the string gadget names a Drawer, then the selected name is appended to the end of the string building a complete path name to the selected name. If the contents of the string gadget is already a path name to a specific file, the last portion of the string (the file name) is replaced. If the selected name is a Drawer name (surrounded by "(" and ")") then the Drawer name is added to the file name string and the named Drawer is scanned.

Type a name into the string gadget

Any time the display is static a complete or partial file name may be typed into the string gadget. Note that typing a <Return> causes the Drawer named (or the Drawer containing the file named) to be scanned. The OK gadget must be selected to complete the Jump Link command.

Type a <Return> into the string gadget

This causes the Drawer named in the string gadget or the Drawer containing the file named in the string gadget to be scanned.

Parent

This causes the last piece of the file name to be removed from the file name in the string gadget and the resulting Drawer is scanned. There is no way to switch disks except by editing the string gadget and typing <Return> to start the scan of the new Drawer name.

Sort Requester Explained

A sort is requested for one of 4 groups of statements. In each case the highest level statement that qualifies is the statement that is re-ordered.

File - the first level statements (branches) are selected to be sorted.

Branch - The first level of subordinate statements (branches) of the designated branch are selected to be sorted.

Group - the statements (branches) in the group are selected to be sorted.

Plex - the statements (branches) in the plex are selected to be sorted.

After the range selection is made the Sort requester appears. This requester is described in the following discussion.

There are three fundamental areas of interest in the Sort Requester. On the left side are two gadgets to select the sort order. On the right side are four gadgets that determine what is to be used as the sort keys in the sort. On the bottom right are two string gadgets to specify the length of the sort key and a string value used as a search string in two of the sorts. Each of the gadgets is explained below.

A-Z

The selected statements will be arranged so that the sort keys are in alphabetically ascending order. A null sort key is the lowest in the sequence.

Z-A

The selected statements will be arranged so that the sort keys are in alphabetically descending order. A null sort key is the lowest in the sequence.

UL case

When selected the case of characters is respected during the sort.

When not selected the case of characters is ignored during the sort.

Labels

The statement labels of the selected statements are used as the sort key. The number of characters of significance in the label is determined by the "Number of Characters" string gadget. Statements without labels are arranged somewhat randomly with respect to each other.

First Chars

The first characters of the statement are used as the sort key. The number of characters of significance is determined by the "Number of Characters" string gadget. Null statements are arranged somewhat randomly with respect to each other.

Content Stmt

With this gadget selected the "Key String" gadget is enabled. The selected statements are each searched for the "Key String" and the characters immediately following the key string are used as the sort key. The number of characters of significance is determined by the "Number of Characters" string gadget.

Content Branch

With this gadget selected the "Key String" gadget is enabled. The first level of subordinate statements to the statements that are selected are searched for the "Key String" and the characters immediately following the key string are used as the sort key. The number of characters of significance is determined by the "Number of Characters" string gadget.

For example: The following statements might be sorted using Content Branch.

(customer name)Customer Name
Serial Number: 123456-12345
Shipdate: 89/01/01
(customer name)Another Customer
Serial Number: 111111-11111
Shipdate: 89/02/01

Should it be desired to sort the customer records (branches) by the Shipdate, then select the Group of customer records and chose Content Branch. The Key String would be "Shipdate: " and the Number of Characters would be 8.

Key String

This specifies the search string for the Content selections. The sort key is the character string that follows the Key String. If the Key String is not located then the sort key is the null string.

Number of Characters 1-27

The number of characters to be included in the sort key. If no value is entered or the value of 0 is entered, then 27 is assumed as the value.

Cancel

Cancel the sort

Sort

Perform the sort based on the selection of the other gadgets. The window will be re-displayed after the sort completes. The anchor of the window after the sort will be the statement "in front of" the statements that were rearranged. If File is selected, the anchor is at the origin of the file. If Branch is selected, the anchor is the branch selected. If Group is selected, the anchor is the statement outside the group closest to the origin of the file. If Plex is selected the Branch of which the plex is subordinate is the anchor of the window.

ARexx Interface

Thinker can be used as a command host in database applications or can drive other applications as a simulated client task. If **Thinker** sends messages to itself then macros are possible but the host interface is designed mostly for database use and not keyboard macros. **Thinker's** public ARexx port is called 'Thinker' (note the mixed case). When **Thinker** is started from the CLI the -NOWINDOW (abbreviated -NOW) option keeps **Thinker** from opening a display window. This option is used when ARexx is the interface of choice. The ARexx program DBTEST.REXX is a small example of using **Thinker** as a database.

ARexx Command Port

Thinker responds to commands sent to the Port "Thinker". When **Thinker** is processing ARexx commands it remembers the last statement that it processed (added or jumped to or fetched) and all subsequent ARexx commands that do not explicitly specify a statement are relative to the last statement processed by an ARexx command (the CURRENT statement). NOTE: If a **Thinker** ARexx command includes a filename (as in "file,label") ONLY **Thinker** documents are recognized. Pictures are not displayed, applications are not launched, ARexx messages are not sent. Use ARexx directly for these functions.

The ARexx port is an independent access to **Thinker** and may be used while the mouse and keyboard are being used. Since the Amiga is a multitasking environment, it is possible that some background task is using **Thinker** as a database while the **Thinker** windows are being used interactively to edit documents. ARexx commands can deal with different files than the ones displayed in any of the **Thinker** windows. There is the concept of the ARexx current file (the last file accessed by any ARexx command) and the ARexx current statement (The last statement accessed by any ARexx command).

Any time that Function Key is used to execute a macro, the ARexx current file is set to the file displayed in the window in which the Function Key was typed, and the ARexx current statement is set to the anchor statement of that window. This makes it possible to write macros that process what is seen on the screen. However, if a macro

is triggered while a background task is using the ARexx port for some database processing, the results can be confusing as the foreground activity may alter the current file and statement that the background task is using.

In many of the following commands there is a free form "data" string. This string is always the last part of commands and **Thinker** usually takes all of the rest of the command starting at the first non-blank character as the "data" string. If the "data" string begins and ends with a single quote (') all of the characters between the quotes is used as the "data" string allowing the "data" string to begin with leading white space. UPDATE STYLES is the exception.

ADD AFTER UP|DOWN|SAME "data" - adds a statement after the CURRENT statement at the specified relative level.

ADD BEFORE UP|DOWN|SAME "data" - adds a statement in front of the CURRENT statement. If the CURRENT statement is the first subordinate statement in a branch, the new statement will become the first subordinate statement in the branch (the DOWN option is an error in this case).

ADD ORIGIN "data" - adds a statement at the origin of the default file. The added statement becomes the new origin.

CLOSE (also QUIT) - terminates **Thinker** updating all modified files.

CLOSEFILE - Closes the file that ARexx is using.

CREATE "filename" - creates a new **Thinker** document. Note that this command does not alter the default file. GET ORIGIN "filename" can be used to access this new file.

CURSOR FETCH - if the cursor is in some statement, returns the cursor offset in that statement. This and the CURSOR SET function are best used after a GET CURSOR operation that fetches the statement in which the cursor is located.

CURSOR SET nnn - if the cursor is in some statement, sets the new cursor offset in that statement.

CUT CURRENT - Copies the **CURRENT** statement onto the **Thinker** clipboard and deletes all the text from the statement (but does not delete the statement).

CUT SELECTED - Cuts the selected text to the **Thinker** clipboard. This makes possible macros that extract information and re-insert it somewhere else in the document. Example below:

```
/* Macro to extract selected text and insert in below */  
options results  
'cut selected'  
'add after down' ""  
'paste current'
```

DELETE CURRENT - Deletes the **CURRENT** statement or branch.

GET CLIPBOARD - the contents of the **Thinker** clipboard are returned.

GET CURRENT - gets the **CURRENT** statement.

GET CURSOR - get the statement in which the cursor is found (only available if a window is open).

GET DOWN - gets any subordinate statement.

GET LABEL FIRST "labelstring" - returns the first statement with the specified label in the default file or if a file is specified in "labelstring" (as in "myfile,funnylabel" in the file specified. If a file is specified, it becomes the default file.

GET LABEL NEXT - gets the next statement with the same label.

GET ORIGIN ["filename"] - Returns the origin statement of the default file or from the filename specified. If a file name is specified, it becomes the default file.

GET PRECEDING - gets the previous statement at the same level.

GET STYLES - gets the style descriptor string for the current statement

This command retrieves a string with a byte for each character of text in the statement. Each byte in the string describes the style of the corresponding text byte in the statement. A return code indicates that there is no style information for the statement. **GET STYLES** retrieves the style information for the "current statement" which is that statement last returned as the result of a **GET** or **JUMP** command. The byte is described below:

0bbffIBU

bb - background pen color

ff - foreground pen color

I - 1 if italic

B - 1 if bold

U - 1 if underlined

The style byte is 0 for plain text.

GET SUCCEEDING - gets the next statement at the same level.

GET UP - gets any parent statement.

INPUT string - Solicits input from the user.

A requester with a string gadget is opened in the ARexx window (most likely the currently active window). The "string" is used as the "title" of the requester (up to 64 characters). The string typed into the string gadget in the requester is returned as the "result" of the command (up to 64 characters).

The **CHANGEALL.THINKR** macro uses this command to solicit the search and replace strings if these strings are not provided as parameters to the **CHANGEALL** macro.

JUMP - same options as **GET**. **JUMP** acts as **GET** in that it returns the statement just as **GET** does, but will also act as a **JUMP** if a window is open.

PASTE CURRENT - Replaces the contents of the **CURRENT** statement with the contents of the **Thinker** clipboard. The clipboard contents are not altered.

POSITION RESTORE n - restores the position saved in the *n*th place (0-9).

POSITION SAVE n - save the **CURRENT** position (file and statement) in 1 of 10 (0-9) places.

REVERT - Discards modifications since the last **SAVE** (or open of the document).

SAVE - Applies modifications made up to this point. If **SAVE** is not use then all modifications are applied with the document is closed (via the **QUIT** or **CLOSE** commands).

SEARCH FIRST "data" - Searches from the **CURRENT** statement until "data" is found. Returns the first statement that contains the string "data".

Note: if a Window is open (not using **Thinker** as a database) the search starts from the anchor of the active Window unless the **CURRENT** statement (as a result of a previous **GET** operation) is not in the same file displayed in the active Window.

SEARCH NEXT - Continues a search and returns the statement with the next occurrence of the string "data" presented on the **SEARCH FIRST "data"** command.

SEARCH REPLACE "data" - If and only if a previous **SEARCH FIRST** or **SEARCH NEXT** command has been successful, the located string is replaced "data".

**SORT FILE|BRANCH|PLEX ASCENDING|DESCENDING
NOCASE LABELS|FIRSTC|CONTENTST|CONTENTBR**

LENGTH nn KEYSTR xxxxxxxx - Sort a portion of the active **Thinker** document. **NOTE:** A document must be made active for **ARexx** access via a **GET** command (there must be a **CURRENT** statement.) The operands parallel the Sort requester gadgets. Operands may be specified in any order except that **KEYSTR** must be last (as the rest of the command line is taken as the **KEYSTR**.)

FILE - The entire active document

BRANCH - The active branch

PLEX - The active plex

ASCENDING - A -> Z order

DESCENDING Z -> A order

NOCASE - when present causes the sort to ignore the case of all characters used in the sort.

LABELS - the labels are used as the sort key

FIRSTC - the first characters of a statement are the key

CONTENTST - The **KEYSTR** string is located in the statement and the characters that follow are used as the sort key

CONTENTBR - As in **CONTENTST** but the subordinate statements are searched rather than that statement that is to be sorted.

LENGTH NN - specifies the length of the sort key. If the **LENGTH NN** operand is omitted 27 is used. The valid range of **NN** is 1 to 27.

KEYSTR - indicates that the rest of the command string is interpreted as a string to locate in the statement or branch. The characters that follow the located string in the text are used as the sort key. **KEYSTR** must be the last operand specified.

TYPE string - treats "string" as if it were typed at the keyboard.

If the cursor is within a statement, the "string" is inserted at the cursor.

UPDATE CURRENT "data" - Replaces the contents of the **CURRENT** statement.

UPDATE STYLES "style bytes" - Replaces the style descriptor string of the current statement. Each byte is as described in **GET STYLES** and the string **MAY NOT** be quoted.

Error Returns:

- 100 - **GET** option not recognized
- 101 - **GET LABEL FIRST** with no label string
- 102 - **GET LABEL FIRST** can't find label
- 103 - **GET LABEL NEXT** no more
- 104 - **GET SUCCEEDING** no more statements
- 105 - **GET PRECEDING** no more statements
- 106 - **GET DOWN** no down statement
- 107 - **GET UP** no up statement
- 108 - **GET ORIGIN 'filename'** File unknown.
- 109 - **GET CURRENT** no current statement

- 110 - **ADD** option not recognized
- 111 - **ADD ORIGIN** no default file
- 112 - **ADD AFTER** no file,Level,or data
- 113 - **ADD BEFORE** as above
- 114 - **ADD <all>** - "data" string truncated

- 120 - **DELETE** option not recognized
- 121 - **DELETE** no current statement

- 130 - **JUMP** option not recognized

- 140 - Poorly formed **ARexx** message

- 150 - Command nesting deeper than 30

- 160 - **CREATE** failed

- 170 - UPDATE command without and data
- 171 - UPDATE no current statement active
- 172 - UPDATE unable to read statement

- 181 - GET CLIPBOARD - clipboard empty

- 190 - CUT CURRENT - "CURRENT" missing
- 191 - CUT CURRENT - no current statement
- 192 - CUT CURRENT - can't read statement

- 200 - PASTE CURRENT - "CURRENT" missing
- 201 - PASTE CURRENT - no current statement
- 202 - PASTE CURRENT - can't read statement

- 210 - SEARCH - subcommand missing
- 211 - SEARCH FIRST - no string
- 212 - SEARCH FIRST - no file active
- 213 - SEARCH FIRST - string over 64 chars
- 214 - SEARCH FIRST - no current statement
- 215 - SEARCH FIRST - string not found
- 216 - SEARCH FIRST - can't to read statement

- 221 - SEARCH NEXT - no previous SEARCH
- 222 - SEARCH NEXT - windows switched!
- 223 - SEARCH NEXT - files switched!
- 224 - SEARCH NEXT - string not found
- 225 - SEARCH NEXT - can't read statement

- 231 - SEARCH REPLACE - no previous SEARCH

- 240 - SORT - no active file or branch
- 241 - SORT - unknown operand keyword
- 242 - SORT - Sort failed (too little memory)
- 243 - SORT - no number following LENGTH
- 244 - SORT - length out of range
- 245 - SORT - Content specified but no KEYSTR
- 246 - SORT - KEYSTR specified but no string
- 247 - SORT - Length of key string > 255
- 248 - SORT - no subordinate statements in branch

- 250 - CURSOR no subcommand

- 251 - CURSOR subcommand not SET or FETCH
- 252 - CURSOR FETCH - no window open
- 253 - CURSOR FETCH - cursor not in any statement
- 254 - CURSOR SET - cursor not in any statement
- 255 - CURSOR SET - no operand
- 256 - CURSOR SET nn - operand not numeric

- 260 - POSITION - no operand
- 261 - POSITION - operand not SAVE or RESTORE
- 262 - POSITION SAVE - no operand
- 263 - POSITION SAVE n - operand not numeric
- 264 - POSITION SAVE n - operand not 0-9
- 265 - POSITION RESTORE - no operand
- 266 - POSITION RESTORE n - operand not numeric
- 267 - POSITION RESTORE n - operand not 0-9
- 268 - POSITION RESTORE n - no saved position

- 270 - TYPE - no string
- 271 - TYPE string - cursor not in window

- 280 - INPUT - no window open (-NOW option)
- 281 - INPUT - function busy

- 410 - GET CURSOR - NOWINDOW option active
- 411 - GET CURSOR - cursor not in window

- 412 - GET STYLES - No current statement
- 413 - GET STYLES - No style information (all plain)

Linking to a Port

The filename portion of a link in a **Thinker** document can name an ARexx port as well as the other types of files supported by **Thinker**. When such a link is selected as the target of a Jump Link command, **Thinker** sends the label portion of the link to the port named in the filename portion of the link as an ARexx Message. This option is only available in a Jump Link command from an open **Thinker** window and not in a Jump Label command from an ARexx command.

If the ARexx libraries can be opened, **Thinker** first looks for a public port with the name of the filename portion of the link

before evaluating whether the filename portion names a file that is a **Thinker** document, IFF picture file, or application program. If a public port with the correct name is found, the label portion is then sent as an ARExx message to that port. Example: Using the Jump Link command on the link "<Myedit,go 1362>" would send the "go 1362" command to the Myedit port. This might result in some editor moving its display to line 1362.

Any result string returned from the Host that processed the command is placed on the **Thinker** clipboard. Use the "Clipboard" Menu option to display the Clipboard.

Because **Thinker** can send messages to its own ARExx port, one could use this feature to extract information from a **Thinker** document and place it on the clipboard. Example: <Thinker,get label first Thinker:otherfile,funnylabel> will put the statement from "Thinker:otherfile" with the label "funnylabel" on the clipboard. The Paste menu option can be used to copy this data into some statement in the current file. Of course, this has a similar effect as using the Copy command.

Macros

When **Thinker** is sent a command on its ARExx port (from any source including **Thinker** itself) that it does not recognize it sends it to the ARExx master task where it is treated as macro request from **Thinker** with **Thinker** as the host.

The CHANGEALL.THINKR file on the distribution disk is one example of a **Thinker** macro. To use the macro Select the JUMP gadget and then the LINK gadget. Enter "Thinker,changeall xxx,yyy" (without the quotes) into the string gadget and press the RETURN key. (NOTE: "REXX,changeall xxx,yyy" works just as well because REXX is the port name of the master ARExx task.) The Confirmation requester "Send Message" will pop up. Confirm the Jump and the macro will run.

The ADVANCE.THINKR file is a macro that advances the cursor to the beginning of the next word.

It may seem odd to use the Jump Link command to execute a macro,

but linking to ARexx ports is an extension of Hypertext. Using links to execute macros is simply linking to **Thinker** itself just like linking to any other application. The link could easily be part of the text as in `<Thinker,changeall xxx,yyy>` and be executed by double clicking on the link.

The Function Keys can be programmed to submit commonly used macros. When selected, the Function Key sends its programmed string to **Thinker's** ARexx port as if a Jump Link `<Thinker,function key string>` command had been executed. See the section "Edit FKeys" in the "Edit" menu discussion (with this document on-line as a Hypertext document the link `<Edit FKeys>` would take you directly to that section.) When a macro is submitted by a Function Key, the ARexx current file is set to the file displayed in the window in which the Function Key was typed and the ARexx current statement is set to the anchor statement of that window.

Thinker macros are first searched for in the drawer that contains the project icon that was used to launch **Thinker**. When **Thinker** is run from the CLI, the working directory is searched first. **Thinker** macros must have the file extension of "THNKR" or no extension at all. **Thinker** macros are also searched for in the global directory "REXX:". One recommendation is to put all **Thinker** macros that are common to all projects in the REXX: directory with the extension "THNKR" and put all macros that are private to a particular project in the same directory (drawer) as the project with no extension or the extension of "THNKR".

What to do If

Disk Full

You will be warned by two requesters. First, AmigaDOS will put up a Disk Full requester. Before responding to this requester you can use a CLI or the Workbench to discard projects and applications from the disk before responding RETRY. If you reply CANCEL, **Thinker** will stop trying to write the modifications and you proceed to the second requester.

If you respond CANCEL to the AmigaDOS disk full requester, **Thinker** will detect a write error and put up a requester. **Thinker's** requester gives no options but forces you to acknowledge the situation. **Thinker** has the changes since the last time "Save" or "Revert" was chosen so nothing is lost, yet. Respond OK and you are back to normal **Thinker** processing. However, the disk file is scrambled because part of it has been updated. You must take some action.

Delete or remove some other project from the disk. Even go as far as deleting **Thinker** from the disk and try Apply Modifications again. This is the same action you could have taken at the AmigaDOS requester before responding RETRY.

As an alternate method of recovery use the Save As requester to save the document on a new disk. **Thinker** will copy the entire file from the source disk to the destination disk and then apply the modifications to the new file. Note that **Thinker** copies the scrambled file from the full disk and then applies the updates. The file on the full disk is still scrambled and at this point there is no hope of recovering it so you might as well delete it. The modifications necessary to recover the scrambled file were applied to the copy of the scrambled file on the new disk.

As a final note. If there are modifications pending to a file when you quit **Thinker** by closing the last window, you only have the option of making room on the disk and selecting RETRY from the AmigaDOS disk full requester. Always keep backup versions of **Thinker** documents!

Memory Low

This problem can be addressed outside the bounds of **Thinker** or from within **Thinker**. Before responding to the Memory Low requester, you can free up memory in RAM by deleting some RAM file.

In most cases you do not have to take any drastic action. Operations that free memory are: "Spell (Free)", "Save", "Revert", "Close".

The operations of "Import", "Export", "Build Index" and "Print" may be incomplete and must be repeated. If the **Thinker** document is modified by the operation then "Revert" will restore it before repeating the operation.

It is a good idea always to use "Save" before using "Import", "Export", "Build Index" or "Print" and "Sort" as this will insure that the maximum memory is available.

Keyboard Menu Command Summary

The following list of Keyboard Command Letters summarizes the Menu options available as keyboard commands.

- 0 - Spacing 0
- 1 - Spacing 1
- 2 - Set Clip to 2
- 5 - Set Clip to 5
- A - Set Clip to 1
- B - move Branch
- C - Cut
- D - Insert Down (immediate)
- E - Jump Link Cursor
- F - Jump Forward
- G - Move Group
- H - Jump Up
- I - Insert After (immediate)
- J - Jump Preceding
- K - Jump Succeeding
- L - Jump Link
- M - Jump Mark
- N - Next (search)
- O - Jump Origin
- P - Paste
- Q - Quit
- R - Jump Return
- S - Save
- T - Jump Down
- U - Insert Up (immediate)
- V - Copy (to clipboard)
- W - Jump Back
- X - Set Statement Lines to 1
- Y - Jump Next
- Z - Set Statement Lines to 99

Tool Types Summary

The following table shows the Tool Type for each option or behavior. The Tool Types marked with a "*" are only valid in the Thinker icon. The Tool Types marked with a "#" are not valid in the Thinker icon. The Save Options menu command will update the project icon with all of the current settings. You can edit the Thinker icon (using INFO) to set defaults that will be active when **Thinker** is started from the CLI or when it is launched from its own icon rather than from a project icon. Settings in the project icon override the default settings when **Thinker** is started from by launching from a project icon. Do not add any spaces to the strings as documented when editing the **Thinker** icon.

Option/Behavior	Tool Type = Default
Clipping	CLIP = 5
Spacing	SPACING = 1
No Jump Confirm	NOCONFIRM = NO
Show Date	DATE = NO
Number of Lines	LINES = 99
Black on White	REVERSE = NO
Auto Jump on INS	AUTO = NO
Read Cache On	CACHE = YES
Flag Forced	FLAG = NO
Hide Labels	HIDE = NO
Error Message Box	MSGBOX = YES
Enable See-Thru	SEETHROUGH = NO
Make local copy	LOCALCOPY = YES
Allow Dup Labels	DUPS = YES
Export Prefix	PREFIX = \in d\
Single Click Jump	SINGLE-CLICK = NO
Allow _[];#{} ^	ALLOW-PUNCTUATION = NO
Disable Typeover	DISABLE-TYPEOVER = NO
Show Anchor Level	SHOW-LEVEL = NO
Reverse Video Cursor	REVERSE-VIDEO-CURSOR = NO
Close Unneeded Files	AUTOCLOSE = NO
Enable Dot Commands	DOTCOMMANDS = NO
Style Delimit Links	COLOR-DELIMITED-LINKS = NO
Replace As Typed	REPLACE-AS-TYPED = NO

Print Line Length**LINELEN = 70****Export bold****BOLD = @****Export underlined****UNDER =****Export italic****ITALIC = |****Function Keys****Fn = string****Display Font****FONT = fontname.size****#Window Descriptions****WINDOW = l,t,w,h/JD/c,s,n:filename***** File Prefixes****prefix: = path***** Max Open Files****MAXFILES = 16***** Lexicon Directory****LEXDIR = Thinker2:**

License Agreement and Guarantee

Poor Person Software grants a nonexclusive license to the Buyer to use **Thinker** and make a reasonable number of copies for backup purposes and ease of use. Poor Person Software's copyright shall be extended to cover these copies and placed on all these copies.

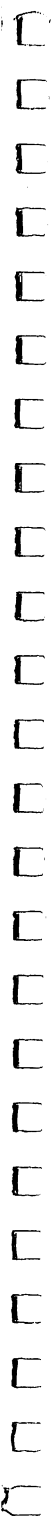
The license does not include the right to re-sell **Thinker** or distribute **Thinker** unless such activity is covered by another agreement.

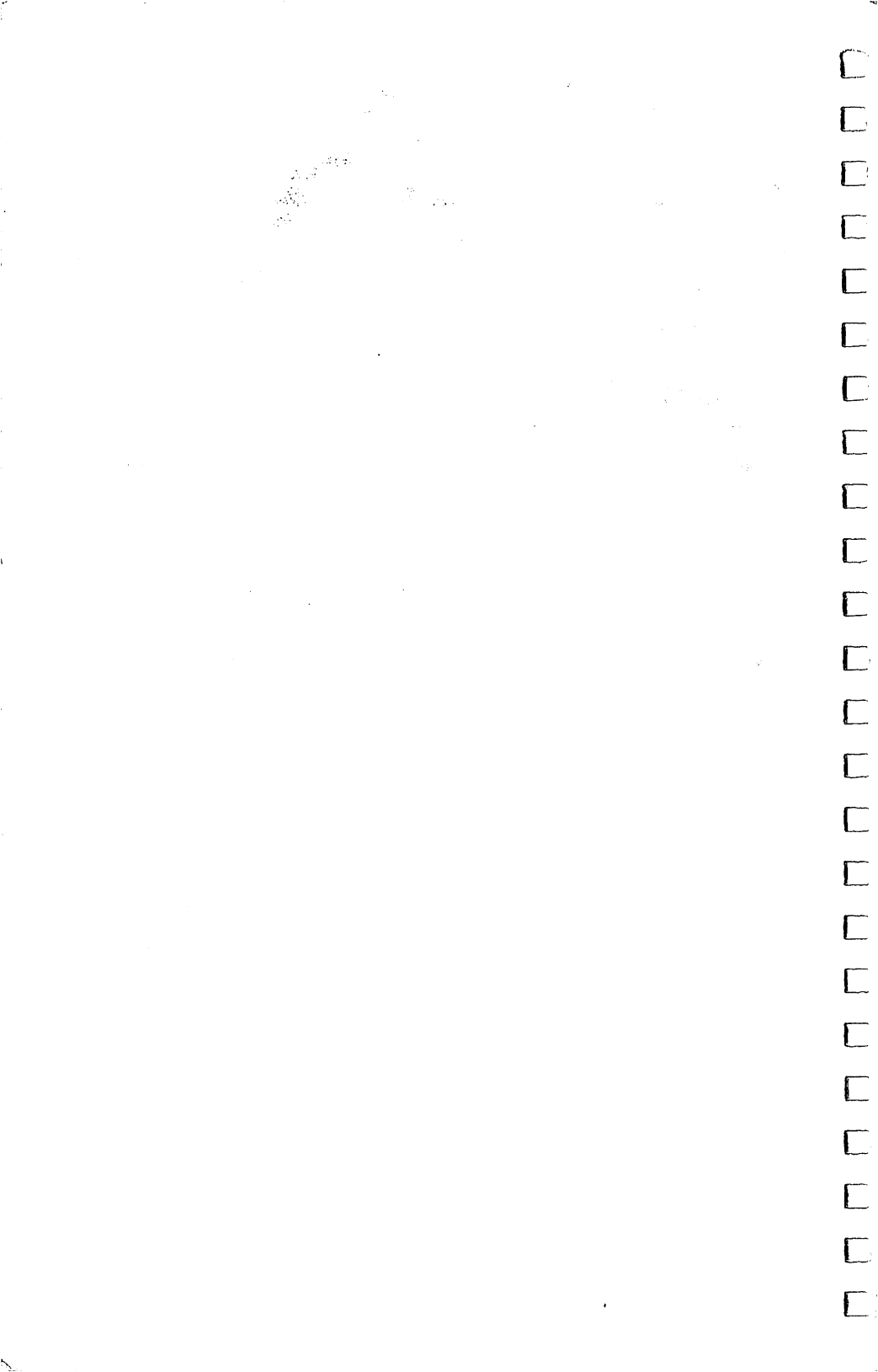
The Buyer agrees to use the supplied software only on a single computer system.

There are no implied warranties. Poor Person Software's liability extends only to replacement of a defective copy or refund of purchase price if the product fails to work as advertised on the Buyer's machine. Buyer uses this software at his or her own risk.

Failure to return the enclosed copy of **Thinker** along with this document and a signed statement to the effect that the Buyer has no additional copies of either the documentation or the machine readable materials implies the acceptance of this agreement by the Buyer.

Thinker may be returned within 30 days of delivery should the Buyer be unsatisfied with the performance or be unable to use **Thinker** due to system incompatibilities. Return all materials and a signed statement that no copies of any material provided by Poor Person Software have been retained.







Poor Person Software

Quality Software, Reasonable Prices

Thinker 2.1

Hypermedia Text Processor for Amiga Workbench 1.3 & 2.0
Requires 512K, best with 1 megabyte

Labels are delimited by "(" and ")" at the beginning of a statement. Statement may have multiple labels.

Structure that is hidden by global view specifications can be expanded locally.

Links to other Thinker documents and ARexx ports are delimited by "<" and ">" characters. Double click to open a window over the labeled section or send command to the port.

Text may be mixed styles and colors. Style changes may serve as link delimiters.

Copy structure elements
Insert new statements
Move structure elements
Delete structure elements
Jump to other sections

The depth of the hierarchy that is visible is controlled by the Clipping level setting.

Sort statements by label or statement contents.

Spelling Checker with 30,000 word expandable dictionary.

Cut and Paste and Edit using the mouse and keyboard. Macros using ARexx.

Thinker displays pictures on a separate screen or in a separate workbench window. Images can be incorporated into documents as statements.

Any word in the text is a potential hypertext link to labeled sections of the same document. Double click the word to move to the labeled section or open a new window over the section.

Double click on ARexx port link to send command directly to another application.

Thinker is a hierarchical text processor. The combination of hypertext, editing, processing, and word processing makes Thinker a combination of word processor, database and authoring system.

Thinker is an outgrowth of the work of Doug Engelbart at SRI during the late 60's and early 70's.

Release 2.1 of Thinker links to text, pictures, ARexx ports or any Workbench Project, Tag, Thinker:logic.drw, and can incorporate images as statements.

ns, Terms) Terms defined

A database is a collection of information ordered in some logical way and often can be randomly.

ARexx command port allows Thinker to be used as a database. Links to ARexx ports allow other applications to process links. Thinker is a multimedia authoring system and idea processor.

(HYPERTEXT) Non-sequential relationship of text with active links that allow the user to read and write documents in logical rather than sequential order.

Unleash your creativity without learning to program. Take advantage of Hypertext and Multimedia without learning a scripting language. All interaction, including adding and deleting links is accomplished using basic text editing operations.

Application Areas:

BBS Message Database

Sort messages by subject
Link messages by reference
Quick scanning with clipping
Selective reading

Study Guides

Add cross reference links
Add commentary
Link to diagrams and pictures

Documentation

Link to definitions
Link to example applications
Link related documents
Link source and documentation

Writing

Outline
Organize
Link related material

Customer Database

Link customer records to
Shipping records
Invoice records
Payment records
Sort by name or content

Storyboard

Outline a story
Link to details
Link to pictures

Organizing Work

Outline work areas
Link to in progress documents
Link directly to applications
for in progress work

Design

Link Descriptions
Specifications